Escuela Universitaria
Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

**ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)**

ANEXOS

Diseño de una estación meteorológica de bajo coste orientada al internet de las cosas

Low-cost Weather Station Design Focused on the Internet of Things

424.21.14

Autor: Jonathan Chamba Benítez

Director: César Asensio Chaves

Fecha: 11 2022

Página intencionadamente en blanco.

INDICE DE CONTENIDO

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

1. ANEXO 1: CÓDIGO FUENTE

En el presente anexo se presentan los diferentes códigos de programación empleados en el desarrollo del proyecto.

1.1. PROGRAMA PRINCIPAL

1.1.1. Cabecera

```
/*--- Librerías ---*/
#include <cmath>          //Librería para operaciones
matemáticas
#include <EEPROM.h>       //Librería para uso de la EEPROM
#include "bsec.h"         //Librería BSEC de BOSCH para el
BME688
#include "ThingSpeak.h"   //Librería para la plataforma
ThingSpeak
#include "WiFi.h"         //Librería para uso de WiFi
#include <SPI.h>          //Librería SPI
#include <Wire.h>         //Librería para uso de I2C


/*-------------------------------------------------------------
-----------------*/
/*--- Configuración de red WiFi ---*/

const char* ssid = " ";             //WiFi ID
const char* password = " ";           //Password
WiFiClient cliente;                           //Variable de
tipo WiFiClient

unsigned long currentMicros = 0;     //Marca de tiempo


/*-------------------------------------------------------------
-----------------*/
/*--- Configuración de ThingSpeak ---*/

unsigned long channelID = ;               //ID de canal
const char* WriteAPIKey = " ";   //Clave de acceso para
canales privados
unsigned long timer_thingspeak = 0;             //Timer para
controlar el tiempo de subida de datos a thinspeak
```

```
unsigned long loopTime_thingspeak = 15000000; //Frecuencia
muestreo 0.0666Hz, 15seg


/*-------------------------------------------------------------
------------------*/
/*--- Conexion ESP32 y BME688 ---*/

#define PIN_SDA 21     //Pin de conexion SDA en ESP32
#define PIN_SCL 22     //Pin de conexión SCL en ESP32
#define LED_BUILTIN 2  //Pin de led integrado en la ESPDUINO-
32

typedef struct{                //Estructura para almacenar los
valores proporcionados por el BME688
  float Temperature;           //Temperatura compensada por el
sensor
  float Pressure;              //Presión
  float Humidity;              //Humedad compensada  por el
sensor
  float GasResistance;         //Resistencia al gas
  float IAQ;                   //Indice de calidad de aire
  float IAQ_Accuracy;          //Precisión del IAQ - el valor
tiene un rango de 0 a 3, donde 0 es baja precision y 3 alta.
La precisión incrementa a mayor número de muestras realizadas
  float Static_IAQ;            //Indice de calidad de aire
para dispositivos estáticos o estacionarios
  float CO2_Equivalent;        //Valor de CO2 equivalente en
ppm
  float BreathVocEquivalent;   //Valor equivalente de COV's
respirados
  int Timer_Trigger;           //Temporizador de activación
del BME688.


}t_sensor_bme688;

t_sensor_bme688 bme688;   //Creación de estructura bme688

/* Configure the BSEC library with information about the
sensor
    18v/33v = Voltage at Vdd. 1.8V or 3.3V
    3s/300s = BSEC operating mode, BSEC_SAMPLE_RATE_LP or
BSEC_SAMPLE_RATE_ULP
    4d/28d = Operating age of the sensor in days
    generic_18v_3s_4d
    generic_18v_3s_28d
    generic_18v_300s_4d
```

```
    generic_18v_300s_28d
    generic_33v_3s_4d
    generic_33v_3s_28d
    generic_33v_300s_4d
    generic_33v_300s_28d
*/
const uint8_t bsec_config_iaq[] = {       //Configuración del
modo de funcionamiento de BME688
#include "config/generic_33v_3s_4d/bsec_iaq.txt"
};

#define STATE_SAVE_PERIOD  UINT32_C(360 * 60 * 1000) // 360
minutes - 4 times a day   //Periodo de tiempo para el
almacenamiento de estado del BME688 en la EEPROM


/*---Declaración de funciones de ayuda del BME688---*/

void checkIaqSensorStatus(void);
void errLeds(void);
void loadState(void);
void updateState(void);
void GetReadingsBME688();


/*---Creacción de objeto de clase BSEC---*/

Bsec iaqSensor;
uint8_t bsecState[BSEC_MAX_STATE_BLOB_SIZE] = {0};
uint16_t stateUpdateCounter = 0;
uint32_t millisOverflowCounter = 0;
uint32_t lastTime = 0;

String Output;



/*-----------------------------------------------------------
------------------*/
/*--- Conexion ESP32 y AS5048A ---*/

static const int SPEED_MAX = 1000000;   //1MHz velocidad de
comunicación del periodo de reloj serial
uint16_t rawData = 0;                   //Variable para
almacenar los valores en bruto leidos por el sensor
int indice = 0;                         //Variable para
recorrer el array de enteros donde se almacena los datos en
bruto
```

```
unsigned long timer_as5048a =
0;                          //Variable para la cuenta de tiempo
transcurrido
unsigned long loopTime_as5048a = 5000; //Frecuencia muestreo
200hz, 5ms o 5000us Los valores seran en microsegundos

/*--- Punteros orientados a objetos de SPI ---*/
SPIClass *vspi = NULL;
uint16_t data = 0b1111111111111111; // Comando de lectura
(0xFFF)


// Definición de pines SPI ESP32
#define VSPI_MISO   19          //Conexion MISO asignado a
pin 19
#define VSPI_MOSI   23          //Conexión MOSI asignado a
pin 23
#define VSPI_SCLK   18          //Conexión CLK asignado a pin
18
#define VSPI_SS     5           //Conexión SS asignado a pin
5


/*--- Parámetros de entrada ---*/
#define num_valores 400
float vGrados[num_valores];
float vDeri[num_valores];
int vec_Frecuencias[120];

/*--- Parámetros derivada numérica ---*/

float degreeData = 0;       //Variable para guardar los
grados medidos
float numDeri = 0;          //Variable para guardar el valor
de la derivada numérica sin procesar
float timeSample = 0.005;   //Tiempo de muestreo en ms para
el calculo de la derivada numérica
float A, B, C;              //Variables para guardar los
valores en orden para operar en la derivada numérica
float outPut = 0;           //Salida de derviada numérica
procesada

/*--- Parámetros de ecuacion de la recta y dirección viento -
--*/
#define SLOPE 2.3712                        //Pendiente de la
recta
```

```
#define FACTOR_GRAD_SEC_TO_KM_H
0.0056548667764616      //Factor de ajuste para convertir
grad/s en km/h
#define OFFSET 6.9151                                    //Offset
float windSpeed = 0;                                  //Variable
que almacena la velocidad del viento
int windDirecction;                                  //Variable que
almacena la dirección del viento
float vSpeed[num_valores];                          //Vector que
almacena las diferentes velocidades para despues promediar
float meanWindSpeed = 0;                              //Variable
que almacena la velocidad promedio


/*--- Parámetros de filtro media movil ---*/
int Mo = 16;
float vec_Filtrado[num_valores];

/*--- Parámetros para almacenar valores de máximos locales --
-*/
int vec_Maximos[num_valores];
float vec_Grados_maximos[num_valores];


/*--- Parámetros para calcular la norma y vector ---*/

float norma = 0.0;
float vec_Generador[120][1];



/*--- Matriz de Proyección para la clasificación ---*/


float D[4][120] ={
{0.0515338401130995,0.125263965938667,0.132662727842823,0,0,0
.198994091764235,0.265325455685647,0.265325455685647,0.464319
547449882,0.0663313639214117,0.198994091764235,0.331656819607
059,0.198994091764235,0.0663313639214117,0.132662727842823,0.
132662727842823,0.132662727842823,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,-0.00385515619372307,-0.0269860933560615,-
0.0424067181309538,-0.00771031238744615,-0.0308412495497846,-
0.0192757809686154,-0.0154206247748923,0,0,0,0,-
0.0131154817930459,-0.00983661134478443,0.053215882128365 8,-
0.0295098340343533,0.0204271776457510,0.0593531125210722,0.06
26319829693336,-0.0221962857124684,-0.0184969047603903,-
0.0110981428562342,0.0478344591610214,0.0626319829693336,-
0.00739876190415613,0,0,0,0,0,0,0,0,0,0,0,0,0,0.0663313639214
```

```
117,0,0,0,0,0.0663313639214117,0.132662727842823,0.1326627278
42823,0.132662727842823,0.0663313639214117,0.198994091764235,
0.0663313639214117,0.132662727842823,0.132662727842823,0.1326
62727842823,0.132662727842823,0.0663313639214117,0.0663313639
214117,0.265325455685647,0.132662727842823,-
0.0115654685811692,0.0277798019841810,0.0902560097118696,-
0.0192757809686154,-
0.0269860933560615,0.128807571649100,0,0.0663313639214117,0,0
,0,0,0,0,0,0,-0.0229520931378303,-0.0229520931378303,-
0.0262309635860918,0.0269849185422740,-
0.00327887044826148,0,-0.0369938095207806,-
0.0295950476166245},
{-0.00470785877466668,-0.0103641906587328,-
0.0106803483618660,0,0,-0.0160205225427990,-
0.0213606967237320,-0.0213606967237320,-0.0373812192665310,-
0.00534017418093300,-0.0160205225427990,-0.0267008709046650,-
0.0160205225427990,-0.00534017418093300,-0.0106803483618660,-
0.0106803483618660,-
0.0106803483618660,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0.000310369097695251,0.00217258368386676,0.0034140600746477
6,0.000620738195390502,0.00248295278156201,0.0015518454884762
6,0.00124147639078100,0,0,0,0,0.160928941387546,0.12069670604
0660,0.155588767206613,0.362090118121979,0.557911120675478,0.
0350501400175201,-
0.00518209532936642,0.000948473109399470,0.000790394257832892
,0.00047423655469973 5,-0.00454977992310010,-
0.00518209532936642,0.000316157703133157,0,0,0,0,0,0,0,0,0,0,
0,0,0,-0.00534017418093300,0,0,0,0,-0.00534017418093300,-
0.0106803483618660,-0.0106803483618660,-0.0106803483618660,-
0.00534017418093300,-0.0160205225427990,-
0.00534017418093300,-0.0106803483618660,-0.0106803483618660,-
0.0106803483618660,-0.0106803483618660,-0.00534017418093300,-
0.00534017418093300,-0.0213606967237320,-
0.0106803483618660,0.000931107293085753,-
0.00223648320398049,-
0.00726628828721823,0.00155184548847626,0.00217258368386676,-
0.0103699792641707,0,-
0.00534017418093300,0,0,0,0,0,0,0,0,0.281625647428206,0.28162
5647428206,0.321857882775092,0.477446649981705,0.040232235346
8865,0,0.00158078851566578,0.00126463081253263},
{-0.00471560574599605,-0.0114622833511001,-
0.0121393073041361,0,0,-0.0182089609562041,-
0.0242786146082721,-0.0242786146082721,-0.0424875755644762,-
0.00606965365206803,-0.0182089609562041,-0.0303482682603401,-
0.0182089609562041,-0.00606965365206803,-0.0121393073041361,-
0.0121393073041361,-
0.0121393073041361,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0.0416976777248991,0.291883744074294,0.458674454973890,0.08
```

33953554497982,0.333581421799193,0.208488388624495,0.16679071
0899596,0,0,0,0,0.00120013259576735,0.000900099446825515,-
0.00486952105630067,0.00270029834047655,-
0.00186918956688229,-0.00543110852660819,-
0.00573114167555003,0.00203107185910797,0.00169255988258998,0
.00101553592955399,-0.00437709376947805,-
0.00573114167555003,0.000677023953035990,0,0,0,0,0,0,0,0,0,0,
0,0,0,-0.00606965365206803,0,0,0,0,-0.00606965365206803,-
0.0121393073041361,-0.0121393073041361,-0.0121393073041361,-
0.00606965365206803,-0.0182089609562041,-
0.00606965365206803,-0.0121393073041361,-0.0121393073041361,-
0.0121393073041361,-0.0121393073041361,-0.00606965365206803,-
0.00606965365206803,-0.0242786146082721,-
0.0121393073041361,0.125093033174697,0.410907123596923,0.4465
35147669754,0.208488388624495,0.291883744074294,0.02955837042
07630,0,-
0.00606965365206803,0,0,0,0,0,0,0,0,0.00210023204259287,0.002
10023204259287,0.00240026519153471,-
0.00246925586476597,0.000300033148941838,0,0.0033851197651799
5,0.00270809581214396},
{0.233362329231269,0.110572485251887,-
0.00814490581833056,0,0,-0.0122173587274958,-
0.0162898116366611,-0.0162898116366611,-0.0285071703641570,-
0.00407245290916528,-0.0122173587274958,-0.0203622645458264,-
0.0122173587274958,-0.00407245290916528,-
0.00814490581833056,-0.00814490581833056,-
0.00814490581833056,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0.000236689570789093,0.00165682699552365,0.002603585278680
03,0.000473379141578187,0.00189351656631275,0.001183447853945
47,0.000946758283156373,0,0,0,0,0.000427395173476135,0.000320
546380107102,-0.00364505773568915,0.000961639140321305,-
0.00257656980199881,0.0553930914193123,0.0552862426259433,0.3
56152173210651,0.296793477675543,0.178076086605326,0.29272102
4766377,0.0552862426259433,0.118717391070217,0,0,0,0,0,0,0,0,
0,0,0,0,0,-0.00407245290916528,0,0,0,0,-0.00407245290916528,-
0.00814490581833056,-0.00814490581833056,-
0.00814490581833056,-0.00407245290916528,-
0.0122173587274958,-0.00407245290916528,-
0.00814490581833056,-0.00814490581833056,-
0.00814490581833056,-0.00814490581833056,-
0.00407245290916528,-0.00407245290916528,-
0.0162898116366611,-
0.00814490581833056,0.000710068712367280,-
0.00170555720127435,-
0.00554132053965054,0.00118344785394547,0.00165682699552365,-
0.00790821624754147,0,-
0.00407245290916528,0,0,0,0,0,0,0,0,0.000747941553583237,0.00
0747941553583237,0.000854790346952271,-

```
0.00279026738873688,0.000106848793369034,0,0.593586955351085,
0.474869564280868}
};


/*---Parámetros resultante que determina la dirección ---*/
float matrizSolu[4][1];
int Direcciones[]={0,90,270,180};


/*--- Funciones auxiliares AS5048A ---*/

void vspiCommand(uint16_t *dato);
void converToDegrees(uint16_t *rawdata, float *degreedata);
void numDerivative(float *a, float *b, float *c, float
*numderi, float *output);
void mediaMovil(float vecDerivada[], int M, float
vecFiltrado[]);
void valMaximos( float vecFiltrado[], int vecMaximos[]);
void gradMaximos(int vecMaximos[], float vecGrados[], float
vecGradMax[], int M);
void vecFrecuencia(float vecGradMax[], int vecFrec[]);
float normaVector(int vecFrec[]);
void vecGenerador(int vecFrec[], float norm, float
vecGen[120][1]);
void multMatrix(float vecProy[4][120], float vecGen[120][1],
float mSol[4][1]);
char calcDirecccion(float mSol[4][1], char direcciones[]);
void printVector(float vector[]);
void printVector_e(int vector[]);
```

1.1.2. Setup

```
void setup() {

  EEPROM.begin(BSEC_MAX_STATE_BLOB_SIZE + 1); // 1st address
for the length
  Serial.begin(115200);



/*-----------------------------------------------------------
------------------*/
/*--------------------PARAMETROS WIFI Y THINGSPEAK----------
------------------*/
/*-----------------------------------------------------------
------------------*/
  WiFi.begin(ssid, password);
```

```
  while(WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.println(".");
  }

  Serial.println("Wifi conectado");
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(100);
  digitalWrite(LED_BUILTIN, LOW);
  delay(100);

  ThingSpeak.begin(cliente);


/*------------------------------------------------------------
------------------*/
/*----------------------PARAMETROS SENSOR BME688------------
------------------*/
/*------------------------------------------------------------
------------------*/
  Wire.begin(PIN_SDA, PIN_SCL);



  iaqSensor.begin(0x77, Wire);
  Output = "\nBSEC library version " +
String(iaqSensor.version.major) + "." +
String(iaqSensor.version.minor) + "." +
String(iaqSensor.version.major_bugfix) + "." +
String(iaqSensor.version.minor_bugfix);
  Serial.println(Output);
  checkIaqSensorStatus();

  iaqSensor.setConfig(bsec_config_iaq);
  checkIaqSensorStatus();

  loadState();

  bsec_virtual_sensor_t sensorList[10] = {
    BSEC_OUTPUT_RAW_TEMPERATURE,
    BSEC_OUTPUT_RAW_PRESSURE,
    BSEC_OUTPUT_RAW_HUMIDITY,
    BSEC_OUTPUT_RAW_GAS,
    BSEC_OUTPUT_IAQ,
    BSEC_OUTPUT_STATIC_IAQ,
    BSEC_OUTPUT_SENSOR_HEAT_COMPENSATED_TEMPERATURE,
    BSEC_OUTPUT_SENSOR_HEAT_COMPENSATED_HUMIDITY,
```

```
    BSEC_OUTPUT_CO2_EQUIVALENT,
    BSEC_OUTPUT_BREATH_VOC_EQUIVALENT,


  };


  iaqSensor.updateSubscription(sensorList, 10,
BSEC_SAMPLE_RATE_LP);
  checkIaqSensorStatus();

  // Print the header
  Output = "Timestamp [ms], raw temperature [°C], pressure
[hPa], raw relative humidity [%], gas [Ohm], IAQ, IAQ
accuracy, temperature [°C], relative humidity [%], Static
IAQ, CO2 equivalent, breath VOC equivalent";
  Serial.println(Output);

/*-----------------------------------------------------------
------------------*/
/*----------------------PARAMETROS SENSOR AS5048A-----------
------------------*/
/*-----------------------------------------------------------
------------------*/



  /*--- Parámetros sensor ---*/
  vspi = new SPIClass(VSPI);  //Inicializacion de la
instancia de la SPIClass adjunta a VSPI
  //vspi -> begin();
  vspi -> begin(VSPI_SCLK, VSPI_MISO, VSPI_MOSI,
VSPI_SS);            //Inicializacion de conexión SPI con los
pines por defecto, SCLK = 18, MISO = 19, MOSI = 23, SS = 5
  pinMode(VSPI_SS, OUTPUT);   //Asignacion de pin 5 como
salida de SPI
  vspi->beginTransaction(SPISettings(SPEED_MAX, MSBFIRST,
SPI_MODE1)); //Configura la velocidad máxima de envio de
datos empezando desde el bit mas significativo, SPIMODE1
obtenido desde el datasheet.


  timer_thingspeak = micros();                    //Inicia la
cuenta desde el inicio de conexión en microsegundos
  timer_as5048a = micros();                       //Inicia la
cuenta desde el inicio de conexión en microsegundos
```

```
}
```

1.1.3. Loop

```
void loop() {

    currentMicros = micros();
    GetReadingsBME688();


    if(currentMicros - timer_as5048a > loopTime_as5048a){
      vspiCommand(&rawData);
      converToDegrees(&rawData, &degreeData);
      vGrados[indice] = degreeData;

      indice++;
      timer_as5048a = currentMicros;

    }



  if(indice == num_valores){
    for(int i = 1; i<num_valores-1; i++){
      A = vGrados[i-1];
      B = vGrados[i];
      C = vGrados[i+1];

      numDerivative(&A, &B, &C, &numDeri,
&outPut);   //Calculo la derivada numérica
      vDeri[i-1] = outPut;  //Guarda el valor de la derivada
en un vector

      //Conversion de grad/s a km/h
      //Serial.print("Salida km/h: ");
      windSpeed = (SLOPE * FACTOR_GRAD_SEC_TO_KM_H) * outPut
+ OFFSET;

      vSpeed[i-1] = windSpeed;


    }


    //Obtencion de velocidad media
    float sumatorio = 0;
```

```
    for(int i = 0; i < num_valores; i++){
      sumatorio += vSpeed[i];

  }
    meanWindSpeed = sumatorio / num_valores;


    f_windDirecction();
    Serial.print("Salida km/h: ");
    Serial.println(meanWindSpeed);

    Serial.print("Dirección viento: ");
    Serial.println(windDirecction);

  }


  if(currentMicros - timer_thingspeak >
loopTime_thingspeak){  //Si se superan los 15 seg, se envian
los datos a thingspeak
    Serial.println("Envio de datos a ThingSpeak");
    SendDataToThingSpeak1();
    timer_thingspeak = currentMicros;

  }


}
```

1.1.4. Funciones de ThingSpeak

```
/*-------------------------------------------------------------
------------------*/
/*-----------------FUNCIONES PARA ENVÍO DE DATOS A THINGSPEAK-
------------------*/
/*-------------------------------------------------------------
------------------*/

void SendDataToThingSpeak1(){  //Valores a enviar si todo
funciona correctamente
    ThingSpeak.setField(1, bme688.Temperature);
    ThingSpeak.setField(2, bme688.Humidity);
    ThingSpeak.setField(3, bme688.Pressure);
    ThingSpeak.setField(4, bme688.Static_IAQ);
    ThingSpeak.setField(5, bme688.BreathVocEquivalent);
    ThingSpeak.setField(6, bme688.IAQ_Accuracy);
```

```
    ThingSpeak.setField(7, meanWindSpeed);
    ThingSpeak.setField(8, windDirecction);

    ThingSpeak.writeFields(channelID,WriteAPIKey);

    Serial.println("Datos enviados a ThingSpeak!");
    //timer2 = currentMillis;

}
```

1.1.5. Funciones BME688

```
/*-------------------------------------------------------------
-----------------*/
/*----------------------FUNCIONES PARA EL SENSOR BME688-----
-----------------*/
/*-------------------------------------------------------------
-----------------*/


void GetReadingsBME688(){     //Funcion que obtiene el valor
del sensor BME688
  unsigned long time_trigger = millis();
  if (iaqSensor.run()) { // If new data is available
    Output = String(time_trigger);
    Output += ", " + String(iaqSensor.rawTemperature);
    Output += ", " + String(iaqSensor.rawTemperature);
    Output += ", " + String(iaqSensor.pressure);
    Output += ", " + String(iaqSensor.rawHumidity);
    Output += ", " + String(iaqSensor.gasResistance);
    Output += ", " + String(iaqSensor.iaq);
    Output += ", " + String(iaqSensor.iaqAccuracy);
    Output += ", " + String(iaqSensor.temperature);
    Output += ", " + String(iaqSensor.humidity);
    Output += ", " + String(iaqSensor.staticIaq);
    Output += ", " + String(iaqSensor.co2Equivalent);
    Output += ", " + String(iaqSensor.breathVocEquivalent);


    Serial.println(Output);
    Serial.println("-----------------------------------------
");

  } else {
    checkIaqSensorStatus();
```

```
  }

  bme688.Temperature = iaqSensor.temperature;
  bme688.Pressure = iaqSensor.pressure;
  bme688.Humidity = iaqSensor.humidity;
  bme688.GasResistance = iaqSensor.gasResistance;
  bme688.IAQ = iaqSensor.iaq;
  bme688.Static_IAQ = iaqSensor.staticIaq;
  bme688.CO2_Equivalent = iaqSensor.co2Equivalent;
  bme688.BreathVocEquivalent =
iaqSensor.breathVocEquivalent;
  bme688.IAQ_Accuracy = iaqSensor.iaqAccuracy;


}

// Helper function definitions
void checkIaqSensorStatus(void)
{
  if (iaqSensor.status != BSEC_OK) {
    if (iaqSensor.status < BSEC_OK) {
      Output = "BSEC error code : " +
String(iaqSensor.status);
      Serial.println(Output);
      for (;;)
        errLeds(); /* Halt in case of failure */
    } else {
      Output = "BSEC warning code : " +
String(iaqSensor.status);
      Serial.println(Output);
    }
  }

  if (iaqSensor.bme680Status != BME680_OK) {
    if (iaqSensor.bme680Status < BME680_OK) {
      Output = "BME680 error code : " +
String(iaqSensor.bme680Status);
      Serial.println(Output);
      for (;;)
        errLeds(); /* Halt in case of failure */
    } else {
      Output = "BME680 warning code : " +
String(iaqSensor.bme680Status);
      Serial.println(Output);
    }
  }
}
```

```
void errLeds(void)
{
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(100);
  digitalWrite(LED_BUILTIN, LOW);
  delay(100);
}

void loadState(void)
{
  if (EEPROM.read(0) == BSEC_MAX_STATE_BLOB_SIZE) {
    // Existing state in EEPROM
    Serial.println("Reading state from EEPROM");

    for (uint8_t i = 0; i < BSEC_MAX_STATE_BLOB_SIZE; i++) {
      bsecState[i] = EEPROM.read(i + 1);
      Serial.println(bsecState[i], HEX);
    }

    iaqSensor.setState(bsecState);
    checkIaqSensorStatus();
  } else {
    // Erase the EEPROM with zeroes
    Serial.println("Erasing EEPROM");

    for (uint8_t i = 0; i < BSEC_MAX_STATE_BLOB_SIZE + 1;
i++)
      EEPROM.write(i, 0);

    EEPROM.commit();
  }
}

void updateState(void)
{
  bool update = false;
  if (stateUpdateCounter == 0) {
    /* Set a trigger to save the state. Here, the state is
saved every STATE_SAVE_PERIOD with the first state being
saved once the algorithm achieves full calibration, i.e.
iaqAccuracy = 3 */
    if (iaqSensor.iaqAccuracy >= 3) {
      update = true;
      stateUpdateCounter++;
    }
  } else {
    /* Update every STATE_SAVE_PERIOD milliseconds */
```

```
    if ((stateUpdateCounter * STATE_SAVE_PERIOD) < millis())
{
      update = true;
      stateUpdateCounter++;
    }
  }

  if (update) {
    iaqSensor.getState(bsecState);
    checkIaqSensorStatus();

    Serial.println("Writing state to EEPROM");

    for (uint8_t i = 0; i < BSEC_MAX_STATE_BLOB_SIZE ; i++) {
      EEPROM.write(i + 1, bsecState[i]);
      Serial.println(bsecState[i], HEX);
    }

    EEPROM.write(0, BSEC_MAX_STATE_BLOB_SIZE);
    EEPROM.commit();
  }
}
```

1.1.6. Funciones AS5048A

```
/*-------------------------------------------------------------
------------------*/
/*----------------------FUNCIONES PARA EL SENSOR AS5048A----
------------------*/
/*-------------------------------------------------------------
------------------*/

/*-------Funcion para calcular la velocidad y direccion del
viento-------------*/

void f_windDirecction(){


  mediaMovil(vDeri, Mo, vec_Filtrado);

  valMaximos( vec_Filtrado, vec_Maximos);
```

```
  gradMaximos(vec_Maximos,vGrados,vec_Grados_maximos, Mo);

  vecFrecuencia(vec_Grados_maximos, vec_Frecuencias);

  norma = normaVector(vec_Frecuencias);

  vecGenerador(vec_Frecuencias, norma, vec_Generador);

  multMatrix(D, vec_Generador, matrizSolu);

  windDirecction = calcDireccion(matrizSolu,  Direcciones);

  if(windDirecction != 0 && windDirecction != 90 &&
windDirecction != 180 && windDirecction != 270){
    windDirecction = vGrados[399];
    Serial.print("Ultimo grado: ");
    Serial.println(vGrados[399]);
  }

  indice = 0;

}



/*---------Lectura de sensor AS5048A-----------------*/

void vspiCommand(uint16_t *dato) {    //Funcion de conexión
SPI entre ESP32 y sensor AS5048A
  //uint16_t data = 0b1111111111111111; //Comando de lectura
(0xFFF)
  //float valor = 0;

  /*--- Uso de SPI similar a como se haria con la API SPI de
Arduino ---*/

  // Envio de comando
  digitalWrite(VSPI_SS, LOW);   //Tirar la SS abajo para
preparar el otro extremo para la transferencia
  vspi->transfer16(data);
  digitalWrite(VSPI_SS, HIGH);  //Tirar la SS alto para
indicar el fin de la transferencia de datos

  // Recibiendo lectura
  digitalWrite(VSPI_SS, LOW);
  *dato = vspi->transfer16(data);  //Almacenamiento de valor
en bruto en variable pasada por puntero
```

```
  digitalWrite(VSPI_SS, HIGH);
  vspi->endTransaction();          //Fin de conexion via SPI
con el sensor

  *dato = *dato & 0b0011111111111111; //Aplicacion de unas
mascara de bits, recogo los 14 bits

}

/*---------Conversión de valores a grados----------------*/

void converToDegrees(uint16_t *rawdata, float *degreedata) {
  *degreedata = (float) * rawdata / 16384.0 * 360.0; //2^14 =
16384

}


/*---------Función que calcula la derivada numérica--------
-------*/

void numDerivative(float *a, float *b, float *c, float
*numderi, float *output) {


  if (*a > *b && *b > *c) { //Si se sigue un orden
desdencente y en sentido horario
    *numderi = (*b - *c) / timeSample; //Aplicacion de la
derivada
    if (abs(*numderi) > 90) {   //Si el resultado es mayor
que 90 grados/seg
      *output = *numderi;
      //Serial.print("Valor 1:");
      //Serial.println(*numderi);  //Imprime el resultado
    } else {          //Caso contrario
      *numderi = 0;         //Imprime un cero
      *output = *numderi;
      //Serial.print("Valor 2:");
      //Serial.println(*numderi);

    }
  } else { //En caso que no se siga un orden descendente
    if (*a > *b && *b < *c) { //Si el ultimo valor ingresado
es mayor que el segundo Y el segundo es menor que el tercero
INDICA discontinuidad
      if (abs(*b - *c) > 350) { //Si el ultimo valor absoluto
de la diferencia entre b y c es superior a 350 grados
        *b = *b + 360; //Realizamos la correción
```

```
        *numderi = (*b - *c) / timeSample; //Aplicamos la
derivada
        if (abs(*numderi) > 90) {    //Si es mayor que 90
grad/seg
          *output = *numderi;
          //Serial.print("Valor 3:");
          //Serial.println(*numderi);    //Imprime el valor
        } else {     //Caso contrario
          *numderi = 0;    //Resultado es cero
          *output = *numderi;
          //Serial.print("Valor 4:");
          //Serial.println(*numderi);
        }
      } else {   //En caso que (b-c) no sea superior a 350
        *numderi = 0;    //Se devuelve un cero
        *output = *numderi;
        //Serial.print("Valor 5:");
        //Serial.println(*numderi);
      }
    } else if (*a < *b && *b > *c) { //Entonces si el valor c
es menor que b, y b es mayor que a INDICA que la
discontinuidad ha pasado de b PERO no nos afecta
      *numderi = (*b - *c) / timeSample; //Aplicamos la
derivada
      if (abs(*numderi) > 90) {    //Verificamos si es mayor
que 90 grad/s
        *output = *numderi;
        //Serial.print("Valor 6:");
        //Serial.println(*numderi);    //Si es mayor
imprimimos
      } else {     //Caso contrario
        *numderi = 0;   //Valor derivada es cero
        *output = *numderi;
        //Serial.print("Valor 7:");
        //Serial.println(*numderi);
      }
    } else if (*a < *b && *b < *c) { //Entonces si a es menor
que b, y b es menor que c INDICA que el sentido de giro es
anti-horario
      *numderi = 0;        //Devolvemos un valor de 0
      *output = *numderi;
      //Serial.print("Valor 8:");
      //Serial.println(*numderi);
    } else {       //Entonces si es otro caso
      *numderi = 0;          //Devolvemos un valor de 0
igualmente
      *output = *numderi;
      //Serial.print("Valor 10:");
```

```
      //Serial.println(*numderi);
    }


  }

  if(*numderi < 50000 ){ //Los 50000 es la puerta de error
cuando existe un cambio de 0 a 360, donde se obtiene una
derivada de 72000.
    *output = *numderi;
    //Serial.print("Valor de salida 1: ");
    //Serial.println(*output);
  }else{
    *numderi = 0;
    *output = 0;
    //Serial.print("Valor de salida 2: ");
    //Serial.println(*output);

    //Serial.println(numderi);
  }
}


/*---------Función que filtra la señal de derivada empleando
una media movil----------------*/

void mediaMovil(float vecDerivada[], int M, float
vecFiltrado[]){
  float sumatorio = 0;

  for(int i = 0; i < num_valores - M ; i++){
    for(int j = 0; j < M; j++){
      sumatorio += vecDerivada[i+j];
      //Serial.print("valor Deri: ");
      //Serial.println(vecDerivada[i+j]);

    }

    vecFiltrado[i] = sumatorio/(float)M;


    //Serial.print("sumatorio: ");
    //Serial.println(sumatorio);
    //Serial.print("Media: ");
    //Serial.println(vecFiltrado[i]);
    sumatorio = 0;
  }
}
```

```
/*---------Función que obtiene los valores máximos locales de
la señal filtrada-----------------*/

void valMaximos( float vecFiltrado[], int vecMaximos[]){
  for(int i = 1; i < num_valores - 1;i++){
    if(vecFiltrado[i] > vecFiltrado[i-1] && vecFiltrado[i] >
vecFiltrado[i+1]){
       vecMaximos[i] = i;
       //Serial.print("Valor maximo: ");
       //Serial.println(vecMaximos[i]);
    }
  }

}


/*---------Función que obtiene los grados correspondientes a
las maximas velocidades-----------------*/

void gradMaximos(int vecMaximos[], float vecGrados[], float
vecGradMax[], int M){
  int contador = 0;

  for(int i = 0; i< num_valores; i++){
    if(vecMaximos[i] != 0){
      vecGradMax[contador] = vecGrados[i+M/2];
      contador+=1;
    }

  }

  contador = 0;

}



/*---------Función que obtiene la frecuencia absoluta de los
grados-----------------*/

void vecFrecuencia(float vecGradMax[], int vecFrec[]){

  for(int i = 0; i < num_valores; i++){
    if(vecGradMax[i] != 0){
      float cociente = vecGradMax[i] / 3.0;
      int posicion = cociente;
```

```
      vecFrec[posicion] +=1;

    }
  }

}

/*---------Función que calcula la norma del vector de
frecuencias absolutas-----------------*/

float normaVector(int vecFrec[]){
  float x = 0.0;
  for(int i = 0; i < 120; i++){
    x += pow(vecFrec[i],2);
    //Serial.print("El valor de x: ");
    //Serial.println(x);
  }
  return sqrt(x);

}


/*---------Función que calcula el vector caracteristico de la
señal recibida-----------------*/

void vecGenerador(int vecFrec[], float norm, float
vecGen[120][1]){

   for(int i = 0; i < 120; i++){
    for(int j = 0; j < 1; j++){
      vecGen[i][j] = float(vecFrec[i])/norm;
      /*Serial.println(vecFrec[i]/norm);
      Serial.println(vecGen[i][j]);
      Serial.print(i);
      Serial.print(",");
      Serial.println(j);*/
    }

      //Serial.println(vecGen[i][0]);

  }
}



/*---------Función que multiplica matrices-----------------*/
```

```
void multMatrix(float vecProy[4][120], float vecGen[120][1],
float mSol[4][1]){

  long double val = 0.0;
  for(int i = 0; i<4; i++){    //

    for(int j = 0; j < 1; j++){  //

      //mSol[i][j] = 0;
      val = 0.0;

      for(int h = 0; h < 120; h++){      //Multiplicacion de
matrices


        val = val +(vecProy[i][h]*vecGen[h][j]);


        //mSol[i][j] = mSol[i][j] +
(vecProy[i][h]*vecGen[h][j]);

        /*Serial.print(float(val) );
        Serial.print(",");
        Serial.print(vecProy[i][h]);
        Serial.print(",");
        Serial.println(vecGen[h][j]);
        Serial.print(i);
        Serial.print(",");
        Serial.print(j);
        Serial.print(",");
        Serial.println(h);*/

      }

      mSol[i][j] = float(val);
      val = 0.0;

      /*Serial.println("Dato:)");
      Serial.println(mSol[i][j]);
      Serial.print(i);
      Serial.print(",");
      Serial.println(j);*/

    }
  }

  val = 0.0;
```

```
}



/*---------Función que calcula la dirección del viento-------
----------*/

int calcDirecccion(float mSol[4][1], int direcciones[]){

  int direccion;
  float a,b,c,d;

  a = mSol[0][0];
  b = mSol[1][0];
  c = mSol[2][0];
  d = mSol[3][0];
  /*Serial.println("Funcion cal:" );
  Serial.println(a);
  Serial.println(b);
  Serial.println(c);
  Serial.println(d);
  Serial.println("---------" );*/

  if(a > b && a > c && a > d){
    direccion = direcciones[0];
  }else if(b > a && b >c && b > d){
    direccion = direcciones[1];
  }else if(c > a && c >b && c > d){
    direccion = direcciones[2];
  }else if(d > a && d >b && d > c){
    direccion = direcciones[3];
  }



  return direccion;
}



/*---------Funciónes que imprimen vectores por el puerto
serie-----------------*/

void printVector_e(int vector[]){
  for(int i = 0; i <120; i++){
    Serial.println(vector[i], 4);
  }
```

```
  Serial.println("-----------------");
  Serial.println("-----------------");
}

void printVector(float vector[]){
  for(int i = 0; i <num_valores; i++){
    Serial.println(vector[i], 4);
  }

}
```

1.2. PROGRAMA ESPDUINO-32, AS5048A Y LABVIEW

```
#include <WiFi.h>          //Libreria para uso de WiFi del
ESP32
#include <WiFiUDP.h>       //Libreria Wifi para conexio UDP
#include <SPI.h>           //Librería SPI



/*--- Conexion ESP32 y LabView ----*/

// Implementacion de WIFI
const char* ssid = "iPhone de Jonathan";    //Nombre de red
WIFI
const char* password = "12345678";          //Contraseña de
red
const char* address = "172.20.10.2";        //Dirección IP de
receptor, Labview
unsigned int localPort =  7000;             //Puerto emisor
const int16_t puerto_remoto = 7001;         //Puerto receptor


WiFiClient Cliente;                         //Variable para
conexión WIFI
WiFiUDP Udp;                                //Variable para
conexión UDP via WIFI

unsigned long timer = 0;                       //Variable
para la cuenta de tiempo transcurrido
unsigned long tiempoFrecuenciaMuestreo = 5000; //Frecuencia
muestreo 200hz, 5ms o 5000us
#define  numero_datos 10                       //Tamaño de
muestras a agregar en el array a enviar via UDP
```

```
uint16_t paquete[numero_datos];                //Array que
almacena las muestras del sensor AS5048A


/*--- Conexion ESP32 y AS5048A ---*/

static const int SPEED_MAX = 1000000;   //1MHz velocidad de
comunicación del periodo de reloj serial
uint16_t rawData = 0;                   //Variable para
almacenar los valores en bruto leidos por el sensor
int indice = 0;                         //Variable para
recorrer el array de enteros donde se almacena los datos en
bruto


/*--- Punteros orientados a objetos de SPI ---*/

SPIClass *vspi = NULL;
uint16_t data = 0b1111111111111111; // Comando de lectura
(0xFFF)

// Definición de pines SPI
#define VSPI_MISO   19          //Conexion MISO asignado a
pin 19
#define VSPI_MOSI   23          //Conexión MOSI asignado a
pin 23
#define VSPI_SCLK   18          //Conexión CLK asignado a pin
18
#define VSPI_SS     5           //Conexión SS asignado a pin
5


void setup() {
  Serial.begin(2000000);        //Inicialización de puerto
serie a 2000000 baudios

  /*--- WIFI ---*/
  WiFi.mode(WIFI_STA);          //Modo de conexión de la
ESP32 como estación (dispositivo que se conecta a un punto de
acceso)
  WiFi.begin(ssid, password);   //Inicialización de conexión
a la red WIFI asignada
  while(WiFi.status() != WL_CONNECTED){ //Condiciona que si
no esta conectado a la red wifi, espere hasta recibir señal
    Serial.println(".");                //imprimiendo un
punto
    delay(500);                         //cada 500ms
  }
```

```
  Serial.print("Wifi conectado: ");        //Imprime que se ha
conectado a la red WIFI
  Serial.println(WiFi.localIP());          //además de indicar
la dirección IP local

  Serial.println("Iniciando conexion con el servidor... ");
  Udp.begin(localPort);                    //Inicializa la
conexión UDP

  timer = micros();                        //Inicia la cuenta
desde el inicio de conexión en microsegundos

  /*--- Parámetros sensor ---*/
  vspi = new SPIClass(VSPI);  //Inicializacion de la
instancia de la SPIClass adjunta a VSPI
  vspi -> begin();            //Inicializacion de conexión
SPI con los pines por defecto, SCLK = 18, MISO = 19, MOSI =
23, SS = 5
  pinMode(VSPI_SS, OUTPUT);   //Asignacion de pin 5 como
salida de SPI
  vspi->beginTransaction(SPISettings(SPEED_MAX, MSBFIRST,
SPI_MODE1)); //Configura la velocidad máxima de envio de
datos empezando desde el bit mas significativo, SPIMODE1
obtenido desde el datasheet.

}



void loop() {

  unsigned long currentMicros = micros();     //Marca de
tiempo

  if(currentMicros - timer >=
tiempoFrecuenciaMuestreo){    //Comparación entre tiempos
requeridos acorde a la frecuencia de muestreo asignada
    vspiCommand(&rawData);                                  /
/Llamada a función de lectura de datos del sensor y paso por
referencia de variable para almacenar valores
    paquete[indice] =
rawData;                              //Almacenamiento de
valor bruto en el array de enteros
    indice++;                                               /
/Incremento de variable indice para almacenar el siguiente
valor
```

```
    timer =
currentMicros;                                    //Actualizaci
ón de variable  timer
  }

  if(indice == numero_datos){                          //Si el
indice equivale al numero de datos por muestra requeridos
      Udp.beginPacket(address,
puerto_remoto);      //Inicializa el envio de paquete via UDP
      Udp.write((uint8_t*)paquete,
numero_datos*sizeof(uint16_t)); //Conversion de paquete de
datos a entero de 8 bits sin signo y escritura del mismo con
su tamaño original de 16 bits
      Udp.endPacket();            //Finalizacion de envio UDP
      indice = 0;                 //Reinicio de indice
    }


}


void vspiCommand(uint16_t *dato){     //Funcion de conexión
SPI entre ESP32 y sensor AS5048A

  /*--- Uso de SPI similar a como se haria con la API SPI de
Arduino ---*/

  // Envio de comando
  digitalWrite(VSPI_SS, LOW);   //Tirar la SS abajo para
preparar el otro extremo para la transferencia
  vspi->transfer16(data);
  digitalWrite(VSPI_SS, HIGH);  //Tirar la SS alto para
indicar el fin de la transferencia de datos

  // Recibiendo lectura
  digitalWrite(VSPI_SS, LOW);
  *dato = vspi->transfer16(data);  //Almacenamiento de valor
en bruto en variable pasada por puntero
  digitalWrite(VSPI_SS, HIGH);
  vspi->endTransaction();          //Fin de conexion via SPI
con el sensor

  *dato = *dato & 0b0011111111111111; //Aplicacion de unas
mascara de bits, recogo los 14 bits

}
```

1.3. MATLAB: ENSAYO 1

```
%Previo al uso del presente código se requiere cargar los ficheros


n = 500;  %Definición de número de muestras a emplear
x = 0:n;  %Vector


%Velocidades (grados/s) y posición (grados)


y = Velocidad_prim_der_sin_filtro(1:n+1);
y = sgolayfilt(y,4,21);
yi = -y; %Inversion de datos para encontrar los minimos


p = Grados(1:n+1);


%Localización de máximos y mínimos
[~, pos_max] = findpeaks(y,"MinPeakDistance",5,"MinPeakProminence",5);
[~, pos_min] = findpeaks(yi,"MinPeakDistance",5,'MinPeakProminence',5);


figure


% Top plot


tiledlayout(3,1)  %División de una ventana para representar tres gráficos


% Top plot


nexttile


plot(x,y);
hold on
plot(pos_max, y(pos_max), 'rv','MarkerFaceColor', 'r');
```

```
plot(pos_min,y(pos_min),'rs','MarkerFaceColor','g');
title('Suroeste - 40 Km/h')
xlabel('Tiempo')
ylabel('Velocidad (g/s)')
hold off



% Bottom plot
nexttile
plot(x,p)
%title('Posición (grados) - Tiempo')
hold on
plot(x,p,'o','MarkerIndices',pos_max,'MarkerFaceColor','red')
plot(x,p,'o','MarkerIndices',pos_min,'MarkerFaceColor','g')
xlabel('Tiempo')
ylabel('Posición (grados)')
hold off


%Localización de posiciones angulares de los máximos y mínimos locales
pos_grados_max = Grados(pos_max);
pos_grados_min = Grados(pos_min);


%Conversión de posiciones angulares de grados a radianes
rad_max = deg2rad(pos_grados_max);
rad_min = deg2rad(pos_grados_min);


%Representación gráfica en un plano polar
nexttile
polarscatter(rad_max,1,'r','filled')
hold on
polarscatter(rad_min,1.5,'g','filled')

title('Posición de máximos y mínimos')


hold off
```

1.4. MATLAB: ENSAYO 2

Se presenta el código a utilizar para la obtención del histograma correspondiente a la dirección Norte. Se emplea el mismo código para las direcciones Este, Oeste y Sur, con la diferencia que se cambia los archivos y nombre de variables.

```
%Creacion de histogramas de las 4 direcciones. Usamos velocidad de 40km/h
%en todos:


%Vector de grados de 0 a 360 espaciado 3 posiciones, 120 grados en total
brks = linspace(0,360,121);


%%%%%%%%%%% NORTE:


%Importación de datos
xN = importdata('Dirección de directorio a utilizar');
xN = xN.data(:,1:3);


xN(:,1) = 0:length(xN(:,1))-1;


% Filtración con función sgolay
xNf = filtro(xN(:,3));


%Gráfica de la señal de velocidad
plot(xN(:,1),xN(:,3),'LineWidth',1)
hold on
plot(xN(:,1),xNf,"Color",'r','LineWidth',1)


title("Comparación señales Norte 40km/h: filtrada vs. sin filtrar")
xlabel('Tiempo')
ylabel('Velocidad (grados/s)')
hold off


%Calculo de máximos y mínimos locales de la señal filtrada
xNM = max1(xNf);
xNm = min1(xNf);


%Obtención de los grados correspondientes a los máximos y mínimos
```

```
aNM = xN(xNM,2);
aNm = xN(xNm,2);

%Cálculo de histograma con maximos
figure
hNM = histogram(aNM,brks);
title('Histograma de valores máximos: Norte a 40km/h')
xlabel('Grados')
ylabel('Frecuencia absoluta')


%Cálculo de histograma con mínimos
figure
hNm = histogram(aNm,brks);
title('Histograma de valores mínimos: Norte a 40km/h')
xlabel('Grados')
ylabel('Frecuencia absoluta')
```

Tras obtener los cuatro histogramas almacenados en diferentes variables, se procede a graficar un histograma en conjunto.

```
%Grafico de histogramas mínimos en conjunto


    figure
    histogram(aNm,brks);


    hold on
    histogram(aEm,brks);
    histogram(aWm,brks);
    histogram(aSm,brks);
    title('Histograma de valores mínimos')
    xlabel('Grados')
    ylabel('Frecuencia absoluta')
    hold off
```

```
legend('Norte','Este','Oeste','Sur','location','best')
```

Después, se obtiene la matriz de proyección.

```
%%%%%%%% MATRIZ DE PROYECCIÓN PARA CLASIFICAR CON MAXIMOS

%Primero se recogen los histogramas anteriores en una
% matriz de generadores M

%Creacion de vectores de cada direccion

Norte = vectorGenerador(hNM)
Este = vectorGenerador(hEM)
Oeste = vectorGenerador(hWM)
Sur = vectorGenerador(hSM)

%Creacion de matriz de generadores
M = [Norte;Este;Oeste;Sur]

%Transpuesta de matriz para colocar cada direccion por cada columna
M = M'

%Creación de imagen de la matriz de generadores
image(M,'CDataMapping','scaled')
title("Huella de valores máximos de las direcciones")
xlabel('Columna de direcciónes: N - E - W - S')
ylabel('Frecuencia')

%Creacion de la matriz de proyección
G = M'* M %Producto escalar, obtecion de matriz métrica
```

```
D = inv(G)*M' %Matriz de proyección
```

Las funciones llamadas en los pasos anteriores se ubican siempre al final de programa en la ventana de Live Editor.

```
function y = min1(x) %Función que detecta los mínimos locales
    n = length(x);
    b = 1:length(x);

    for i = 2:(n-1)
        if(x(i) < x(i-1) && x(i) < x(i+1))
            y(i) = b(i);
        end

    end
    y(y==0)=[];
end

function y = max1(x) %Función que detecta los máximos locales
    n = length(x);
    b = 1:length(x);

    for i = 2:(n-1)
        if(x(i) > x(i-1) && x(i) > x(i+1))
            y(i) = b(i);
        end

    end
    y(y==0)=[];
end

function y = filtro(x) %Función que aplica un filtro Savitzky–Golay de orden 4
    orden = 4;
```

```
        longitudFrame = 21;
        y = sgolayfilt(x,orden,longitudFrame);
    end



    function y = vectorGenerador(h) %Funcion que obtiene el vector generador de cada histograma
            hc = h.Values;
            nr = norm(hc);
            y = hc/nr;
    end
```

2. ANEXO 2: DIAGRAMAS DE ACTIVIDAD UML

Este anexo incluye los diagramas de flujo que describen el código fuente.

2.1. PROGRAMA PRINCIPAL

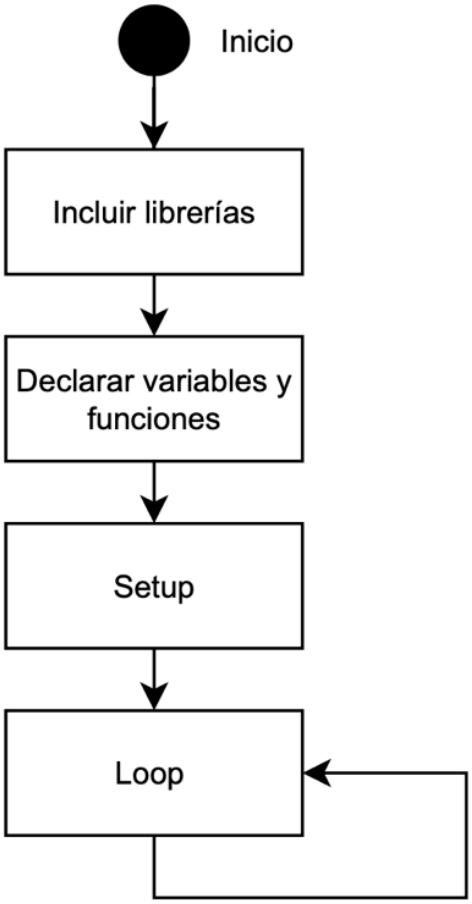


Ilustración 1. Diagrama UML: Funcionamiento conceptual del software del prototipo

2.1.1. Setup

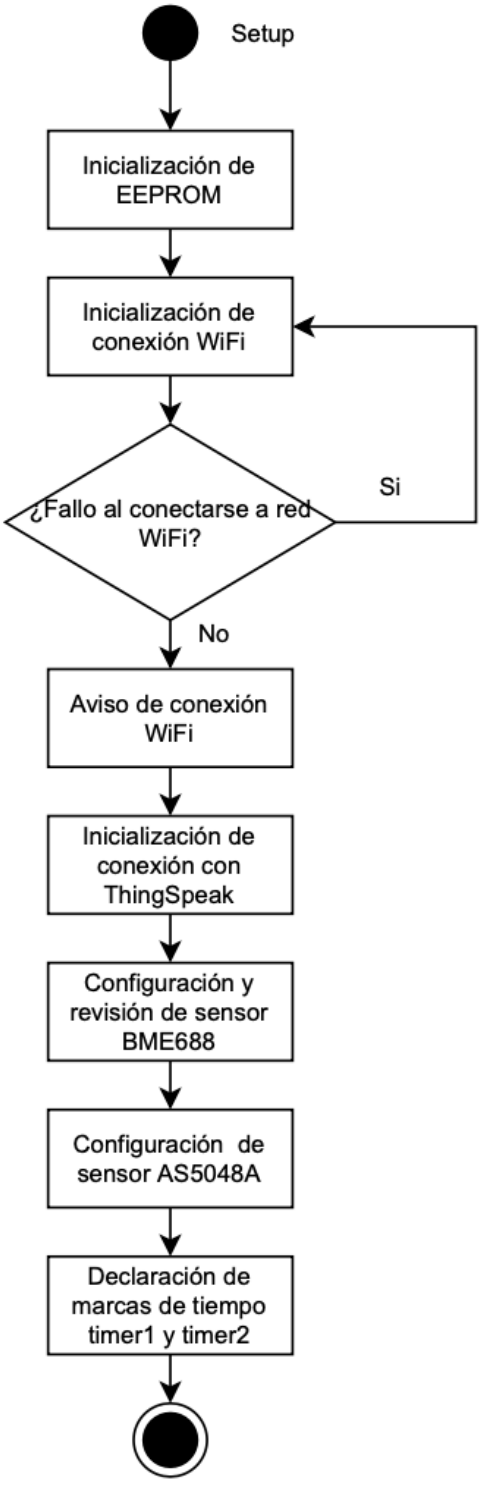


Ilustración 2. Diagrama UML: Setup del código de programación general

2.1.2. Loop

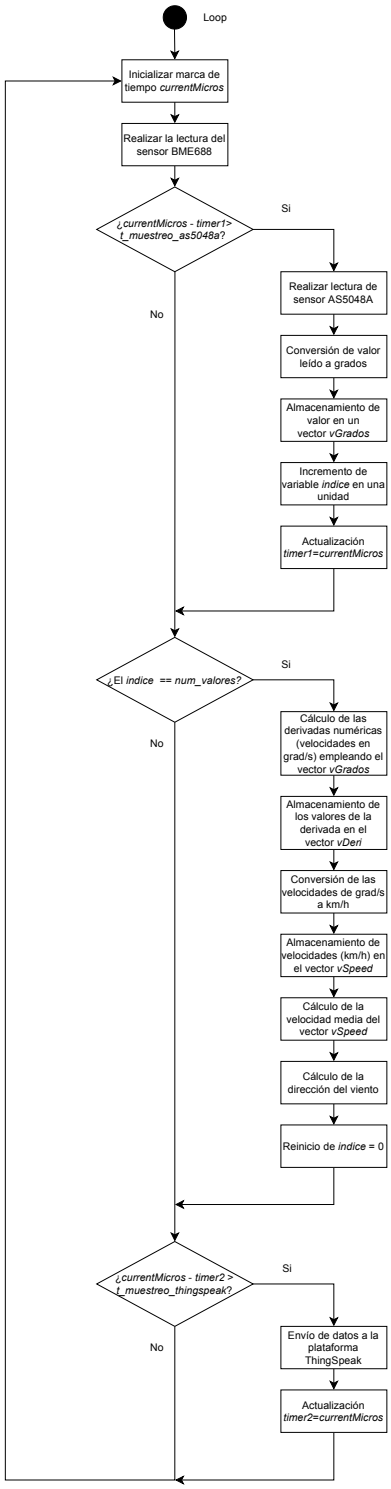


Ilustración 3. Diagrama UML: Loop del código de programación general

2.2. ESPDUINO-32 Y AS5048A

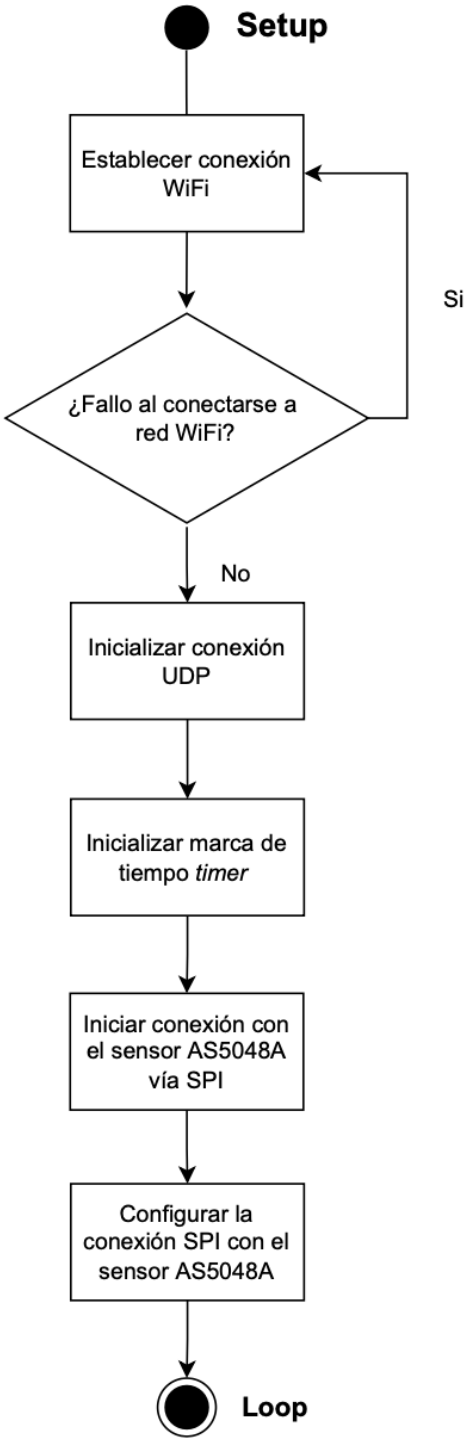


Ilustración 4. Diagrama UML: Setup de la comunicación ESP32, AS5048A y LabVIEW

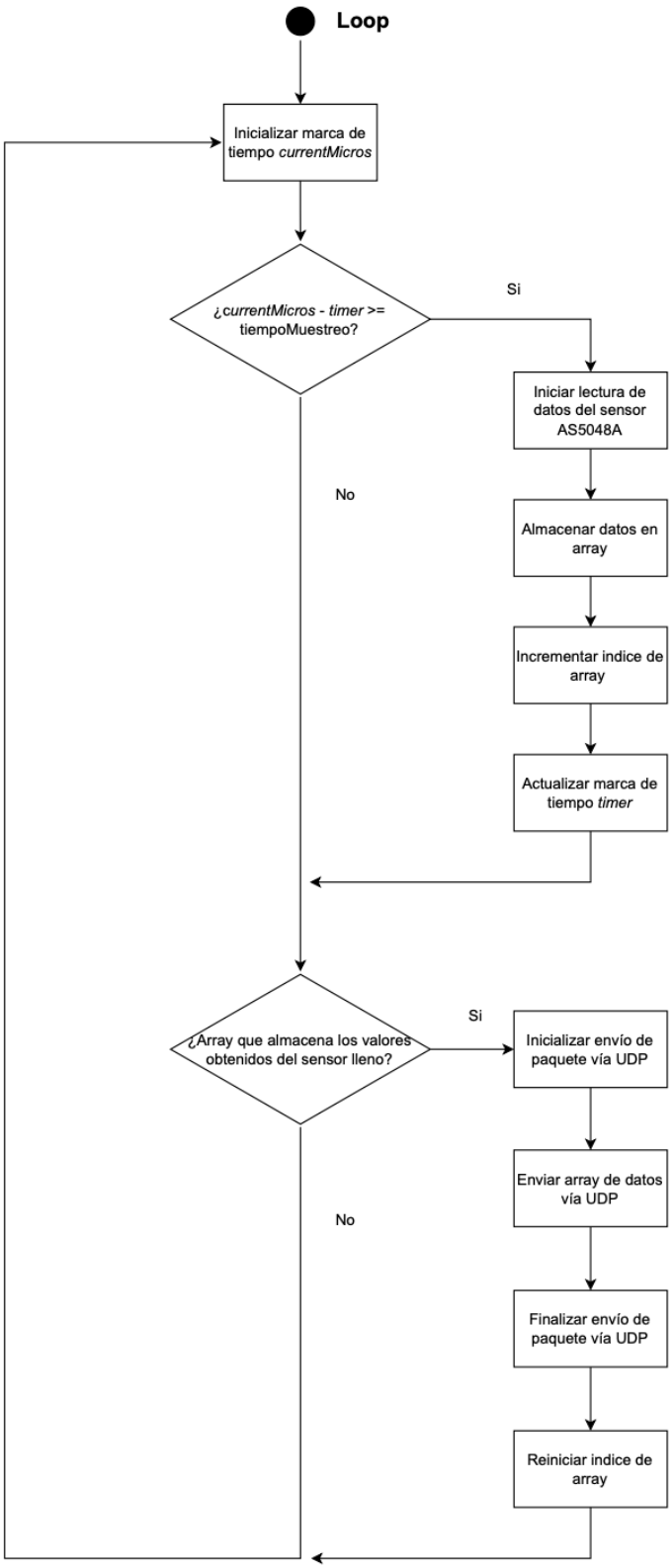


Ilustración 5. Diagrama UML: Loop de la comunicación de ESP32, AS5048A y LabVIEW

2.3. ESPDUINO-32 Y LABVIEW

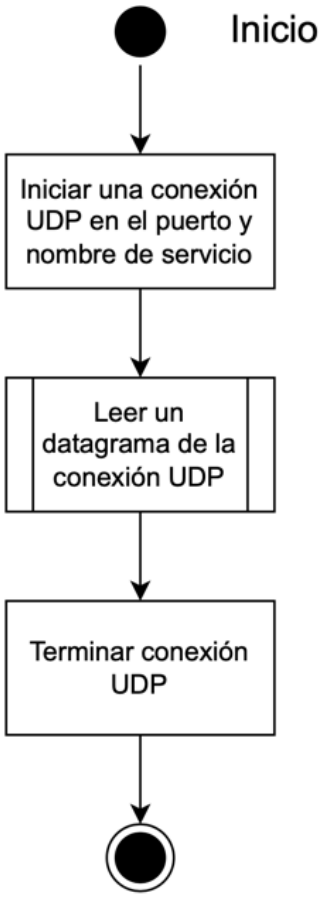


Ilustración 6. Diagrama UML: funcionamiento conceptual de LabVIEW

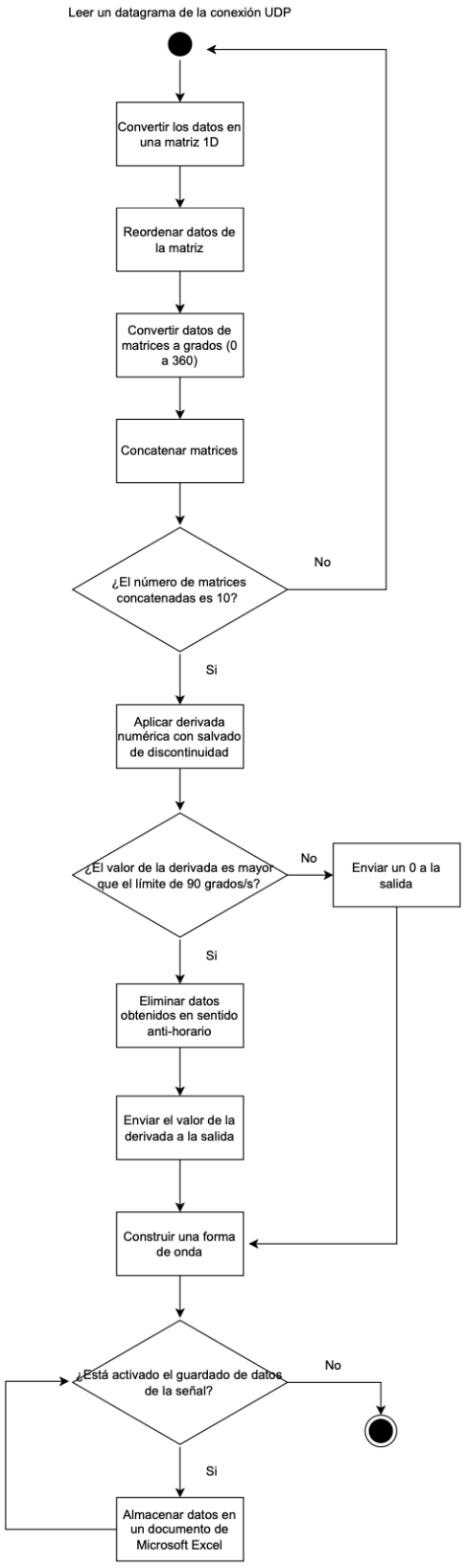


Ilustración 7. Procesamiento de datos en LabVIEW

2.4. CÓDIGO MATLAB

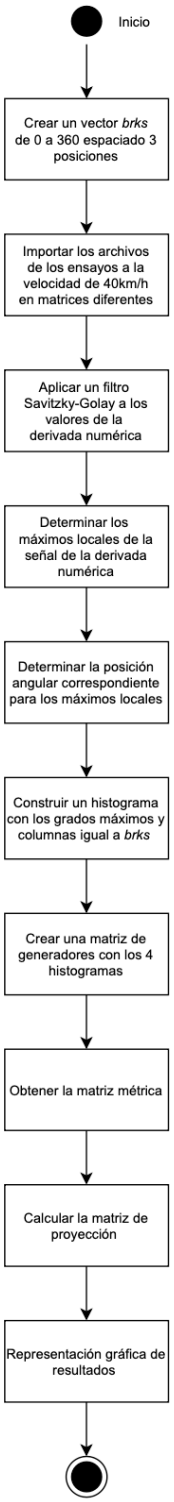
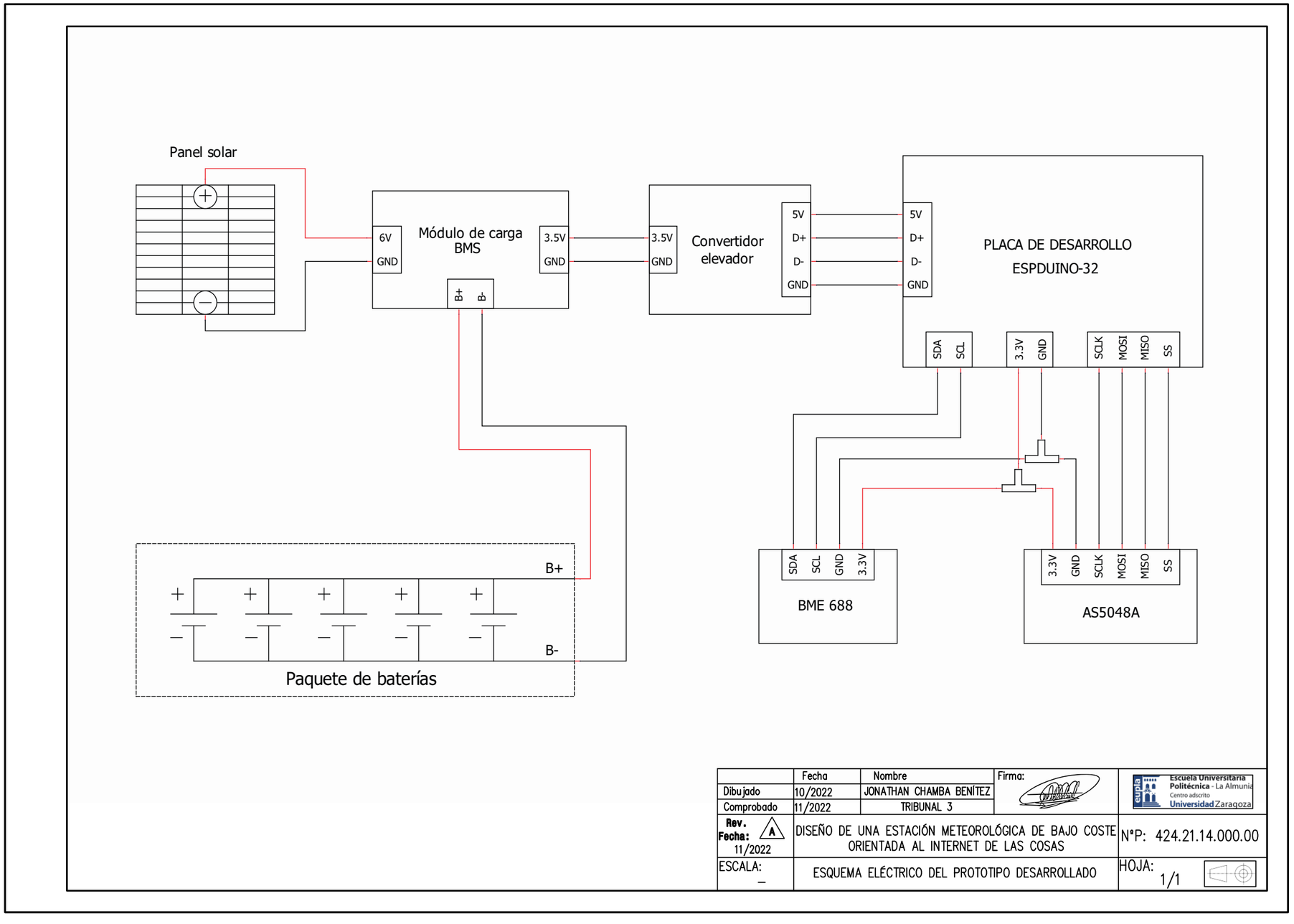


Ilustración 8. Creación de sistema de clasificación en Matlab.

3. ANEXO 3: ESQUEMA ELÉCTRICO



4. Anexo 4: Modificación de IDE Arduino

En este apartado se presentan los pasos a seguir para habilitar la lectura de las librerías BSEC en el entorno de desarrollo de Arduino.

El primer paso es identificar la ubicación del archivo *platform.txt* perteneciente para la configuración de la ESP32.

Para ello en el cuadro de preferencias, se necesita abrir el archivo que se indica a continuación.

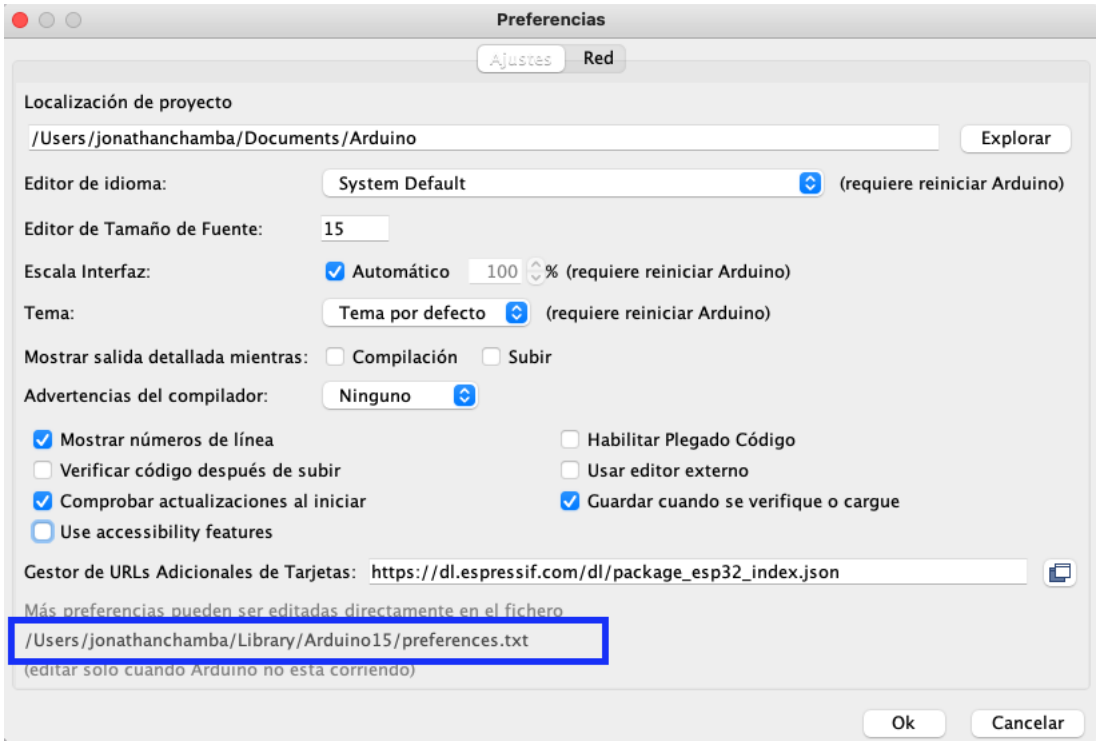


Ilustración 9. Ventana de preferencias del IDE de Arduino

Desplegada la ventana emergente, se procede a abrir la carpeta correspondiente a la ESP32.

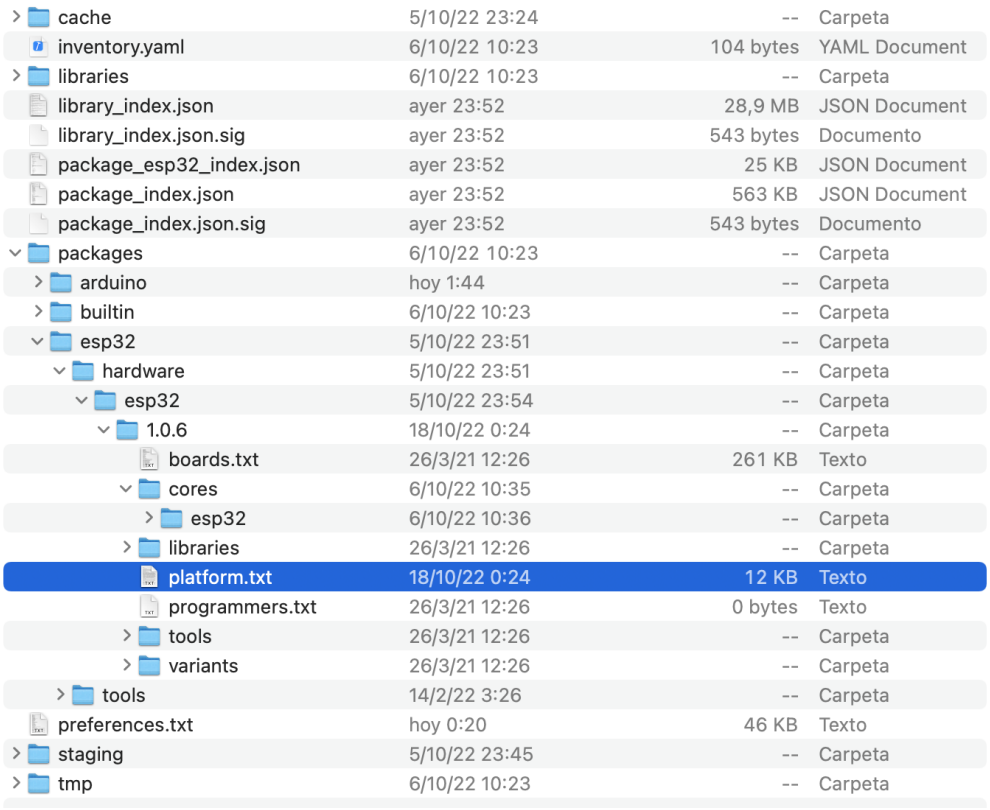


Ilustración 10. Ubicación de archivo platform.txt de la ESP32 en la carpeta Arduino15

Una vez abierto el archivo con un procesador de texto, se debe identificar que la siguiente línea se encuentra definida.

Compiler.libraries.ldflags=

Posteriormente se identifica la siguiente parte de código.

Combine gc-sections, archives, and objects

recipe.c.combine.pattern="{compiler.path}{compiler.c.elf.cmd}" {compiler.c.elf.flags} {compiler.c.elf.extra_flags} {compiler.libraries.ldflags} -Wl,--start-group {object_files} "{archive_file_path}" {compiler.c.elf.libs} {build.extra_libs} -Wl,--end-group -Wl,-EL -o "{build.path}/{build.project_name}.elf"

Y se lo modifica tal como se indica a continuación.

Combine gc-sections, archives, and objects

*recipe.c.combine.pattern="{compiler.path}{compiler.c.elf.cmd}" {compiler.c.elf.flags} {compiler.c.elf.extra_flags} {compiler.libraries.ldflags} -Wl,--start-group {object_files} "{archive_file_path}" {compiler.c.elf.libs} {build.extra_libs}**{compiler.libraries.ldflags}** -Wl,--end-group -Wl,-EL -o "{build.path}/{build.project_name}.elf"*

Se observa en negrita la modificación realizada. Se ha implementado el *Compiler.libraries.ldflags* después la directiva *-strat group.*

Realizado los cambios se procede a guardar el archivo y reiniciar el IDE de Arduino.

5. APÉNDICE 1: GRÁFICOS DE ENSAYO 1

5.1. DISEÑO DE ROTOR 1

Se presentan los diferentes gráficos obtenidos del ensayo 1 (versión 1).

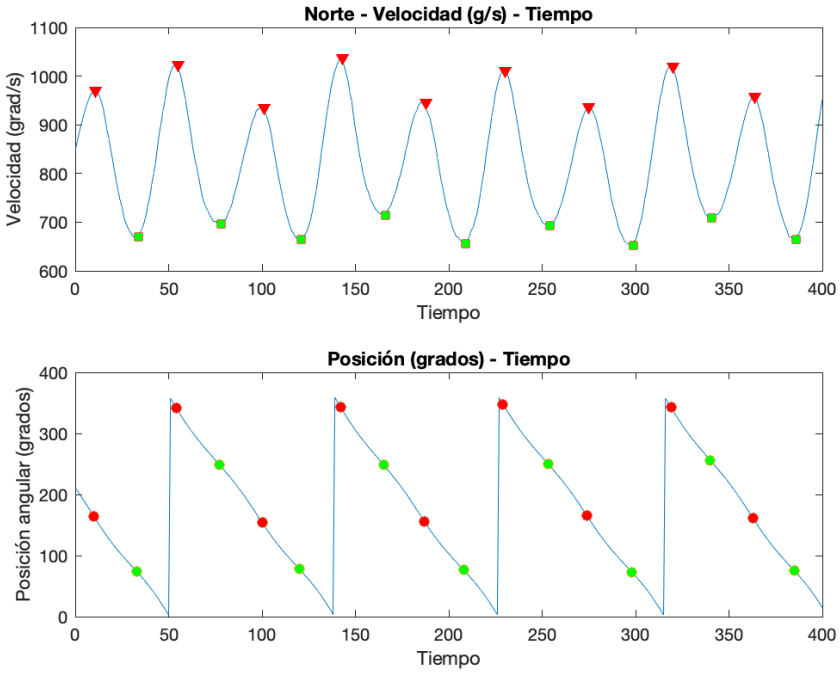


Ilustración 11. Diseño de rotor 1: máximos y mínimos locales de dirección Norte (v1)

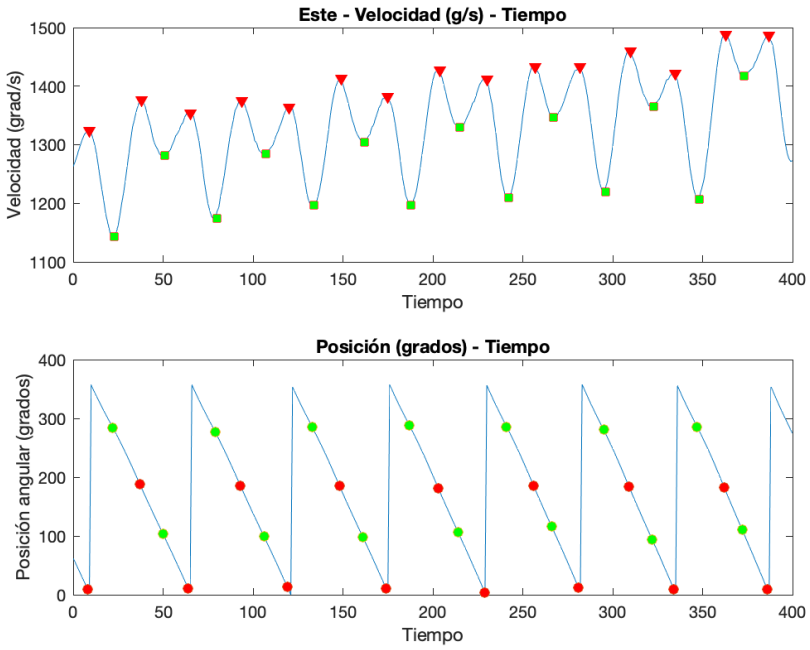


Ilustración 12. Diseño de rotor 1: máximos y mínimos locales de dirección Este (v1)

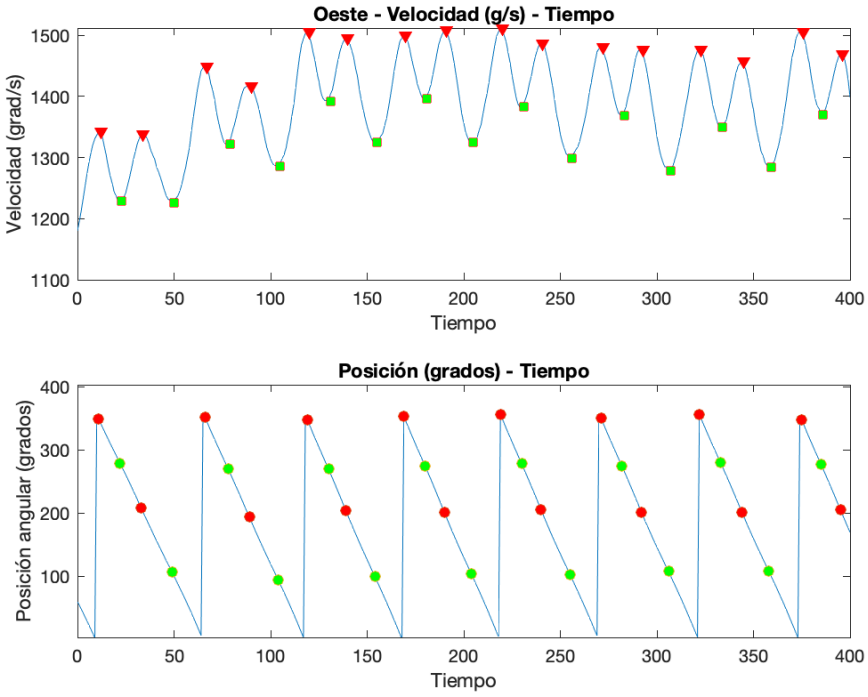


Ilustración 13. Diseño de rotor 1: máximos y mínimos locales de dirección Oeste (v1)

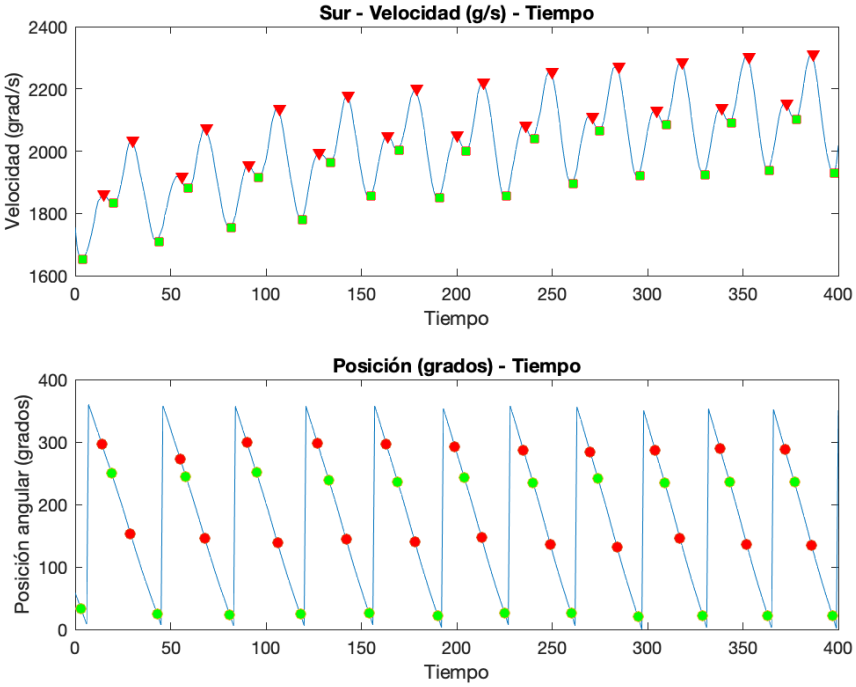


Ilustración 14. Diseño de rotor 1: máximos y mínimos locales de dirección Sur (v1)

Los resultados obtenidos al repetir el ensayo con el diseño 1 son los siguientes.

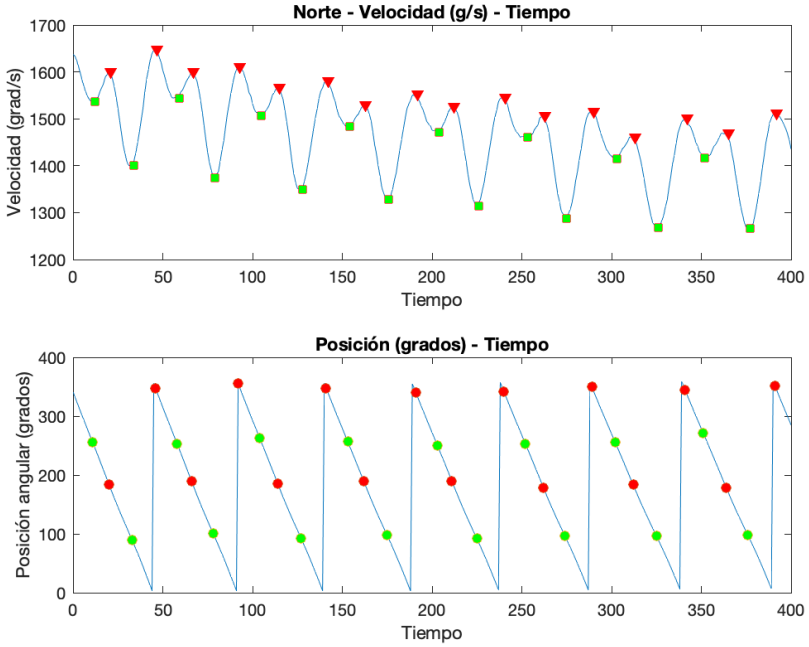


Ilustración 15. Diseño de rotor 1: máximos y mínimos locales de dirección Norte (v2)

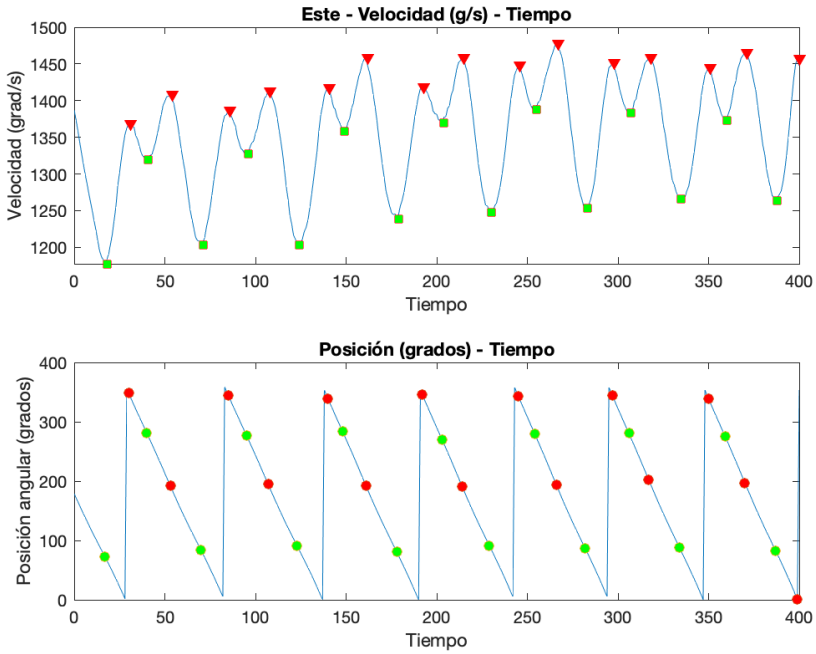


Ilustración 16. Diseño de rotor 1: máximos y mínimos locales de dirección Este (v2)

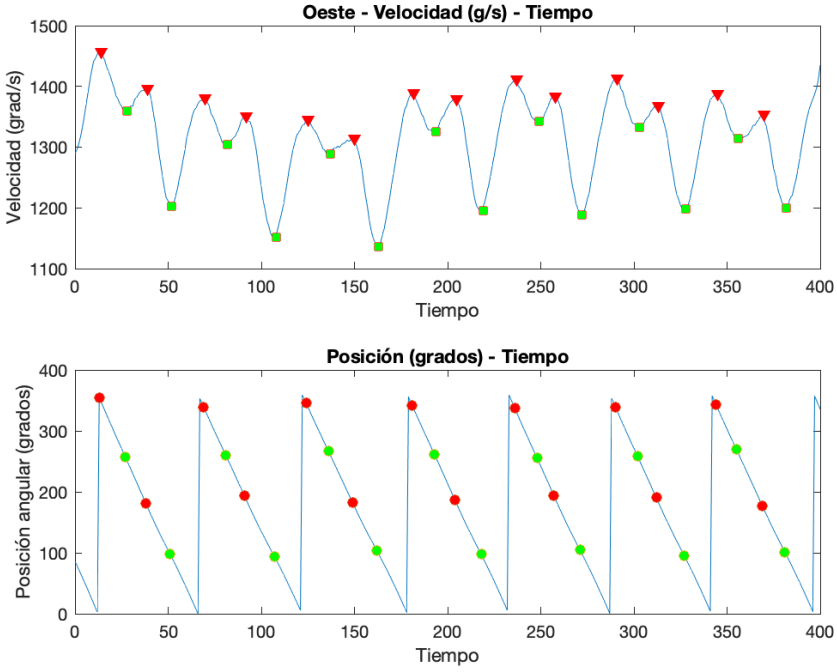


Ilustración 17. Diseño de rotor 1: máximos y mínimos locales de dirección Oeste (v2)

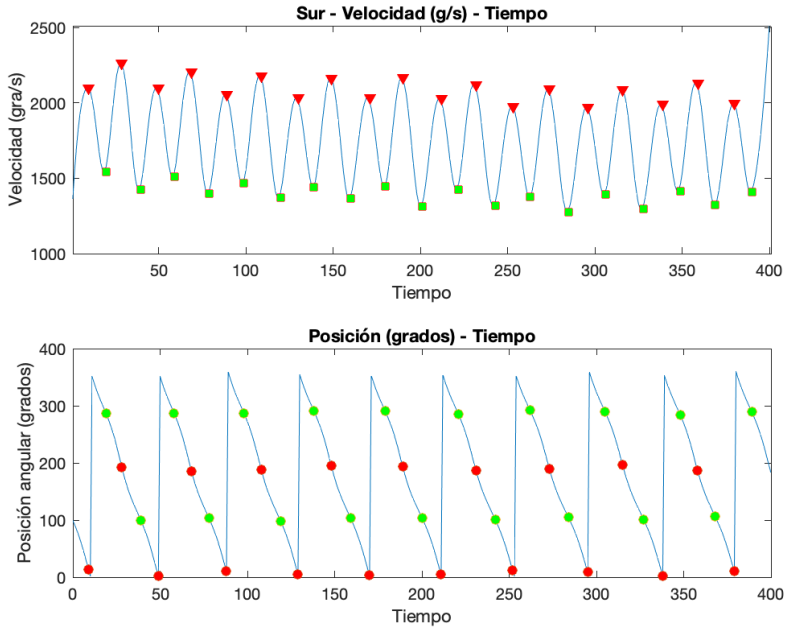


Ilustración 18. Diseño de rotor 1: máximos y mínimos locales de dirección Sur (v2)

5.2. DISEÑO DE ROTOR 2

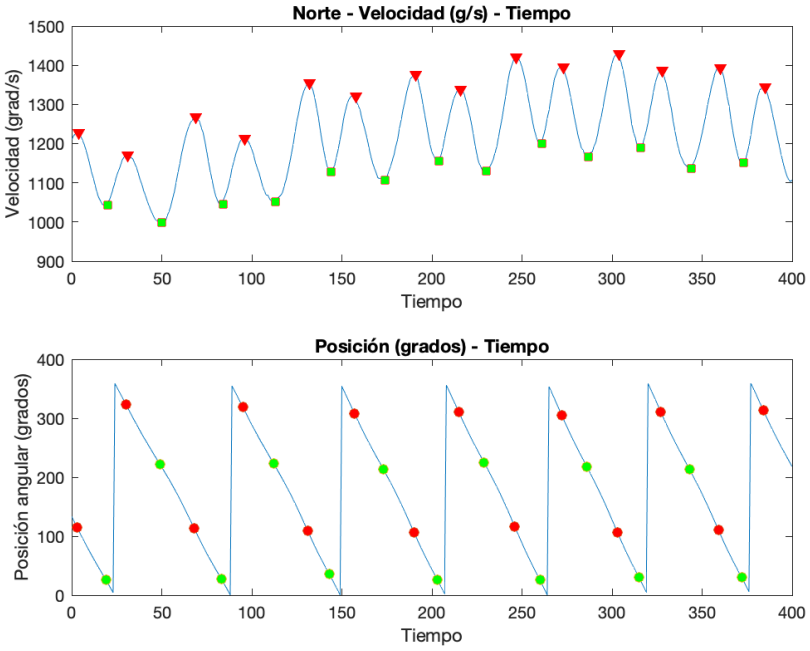


Ilustración 19. Diseño de rotor 2: máximos y mínimos locales de dirección Norte (v1)

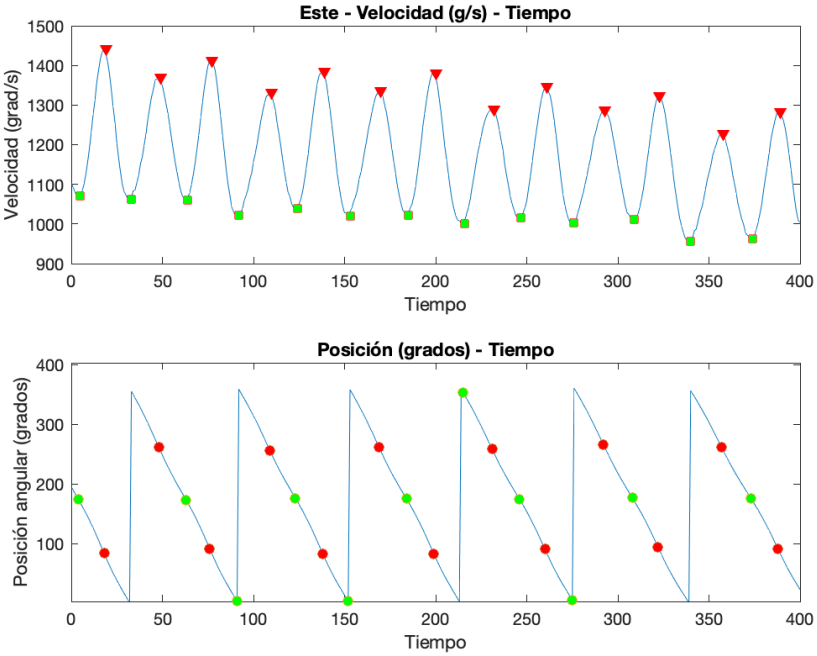


Ilustración 20. Diseño de rotor 2: máximos y mínimos locales de dirección Este (v1)

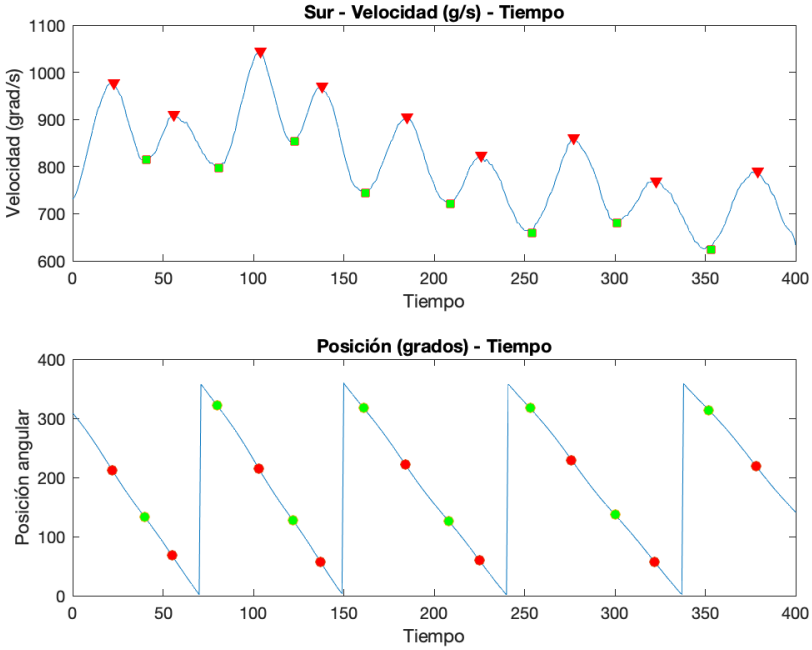


Ilustración 21. Diseño de rotor 2: máximos y mínimos locales dirección Sur (v1)

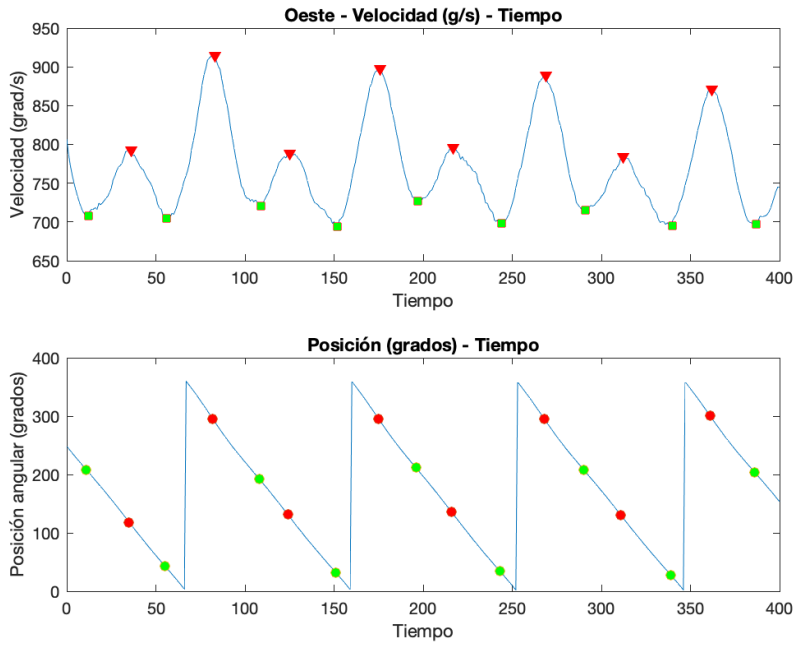


Ilustración 22. Diseño de rotor 2: máximos y mínimos locales dirección Oeste (v1)

Los resultados obtenidos al repetir el ensayo 1 con el diseño de rotor 2 son los siguientes:

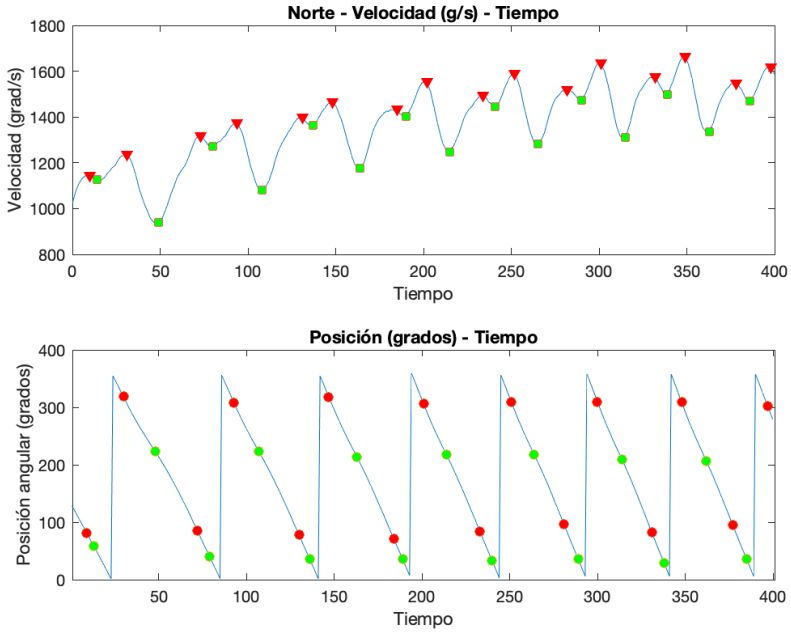


Ilustración 23. Diseño de rotor 2: máximos y mínimos locales dirección Norte (v2)

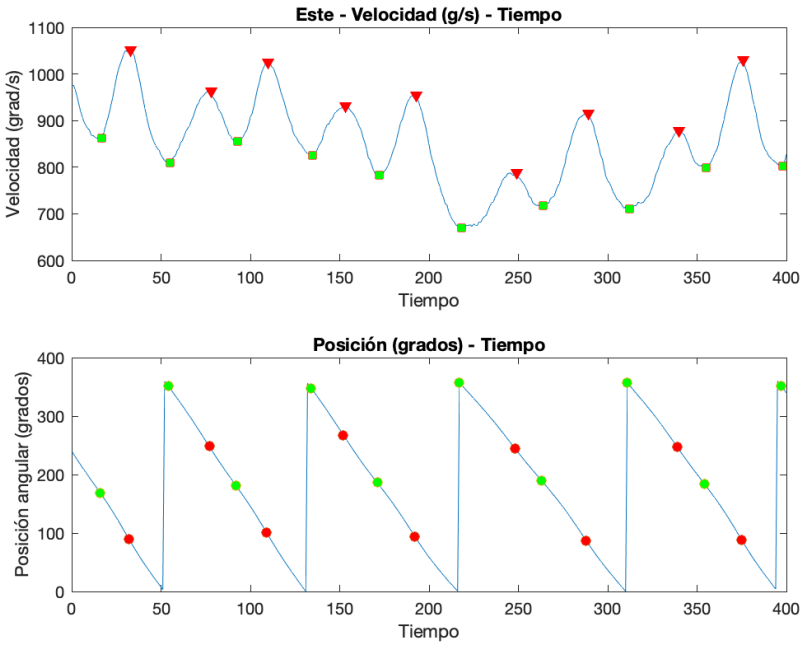


Ilustración 24. Diseño de rotor 2: máximos y mínimos locales dirección Este (v2)

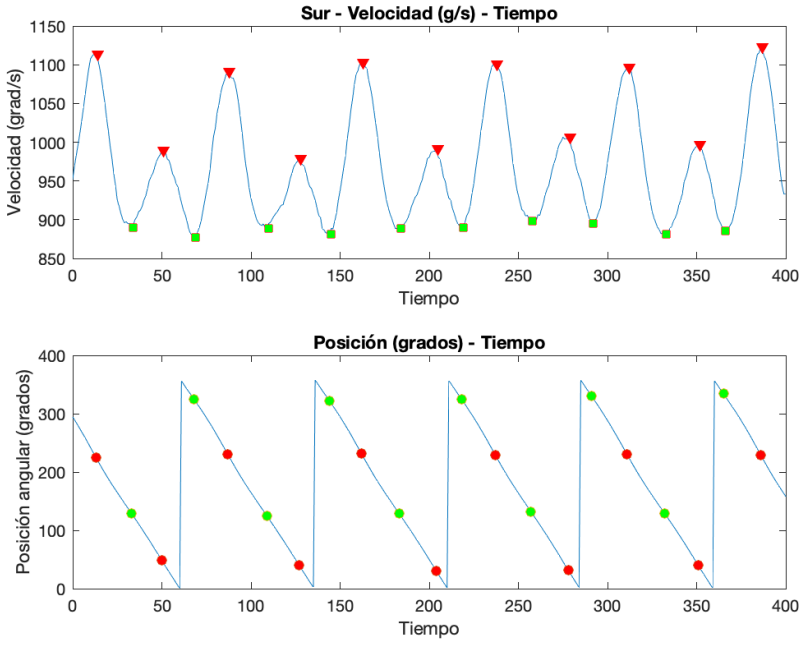


Ilustración 25. Diseño de rotor 2: máximos y mínimos locales dirección Sur (v2)

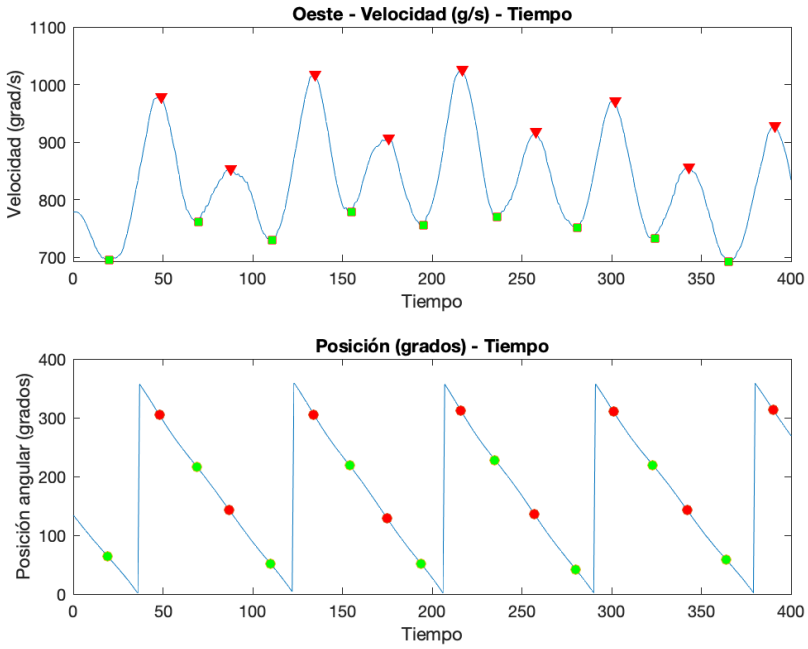


Ilustración 26. Diseño de rotor 2: máximos y mínimos locales dirección Oeste (v2)

6. APÉNDICE 2: DATOS EMPLEADOS EN EL ENSAYO 2

6.1. TABLAS UTILIZADAS EN LA CALIBRACIÓN DE LA VELOCIDAD

Tabla 1. Datos dirección Norte, velocidad de vehículo 20km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	20,340	940,43
2	19,900	918,46
3	20,480	918,46
4	21,130	900,88
5	21,920	922,85
6	20,980	936,04
7	20,190	931,64
8	19,260	918,46
9	18,860	936,04
10	18,360	962,40
11	18,680	922,85
12	19,360	922,85

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
13	19,690	1001,95
14	20,160	975,59
15	20,590	962,40
16	20,800	958,01
17	21,130	971,19
18	21,600	971,19
19	21,380	971,19
20	21,060	975,59
21	20,300	975,59
22	20,050	1010,74
Promedio	20,271	951,32

Tabla 2. Datos dirección Norte, velocidad de vehículo 40 km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	37,720	1823,730469
2	37,720	1819,335938
3	37,400	1828,125
4	36,030	1788,574219

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
5	36,320	1801,757813
6	36,030	1806,152344
7	36,070	1819,335938
8	36,320	1841,308594
9	36,140	1819,335938
10	36,680	1823,730469
11	37,580	1894,042969
12	38,410	1854,492188
13	38,410	1889,648438
14	38,160	1929,199219
15	37,650	1924,804688
16	36,930	1929,199219
17	36,500	1916,015625
18	36,250	1964,355469
19	35,490	1951,171875
20	35,170	1942,382813
21	35,560	1942,382813
Promedio	36,788	1871,861049

Tabla 3. Dirección Norte, velocidad de vehículo 60km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	54,210	3322,265625
2	56,160	3357,421875
3	56,840	3304,6875
4	54,610	3388,183594
5	54,100	3304,6875
6	53,240	3313,476563
7	51,870	3269,53125
8	50,290	3243,164063
9	49,820	3238,769531
10	50,580	3194,824219
11	50,360	3234,375
12	50,400	3212,402344
13	50,970	3155,273438
Promedio	52,573	3272,235577

Tabla 4. Dirección Este, velocidad de vehículo 20 km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	18,750	997,5585937
2	19,650	997,5585937
3	19,720	997,5585937
4	19,580	1050,292969
5	19,400	993,1640625
6	18,860	988,7695312
7	18,820	1019,53125
8	18,820	975,5859375
9	18,970	1006,347656
10	18,820	958,0078125
11	18,680	1001,953125
12	18,640	979,9804687
13	18,860	1028,320313
14	18,930	966,796875
15	18,750	979,9804687
16	18,680	997,5585937
17	18,250	979,9804687

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
18	18,140	962,4023437
Promedio	18,907	993,4082031

Tabla 5. Dirección Este, velocidad de vehículo 40km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	30,560	2100,585938
2	30,450	2285,15625
3	28,330	2452,148438
4	28,260	2487,304688
5	28,000	2518,066406
6	27,610	2469,726563
7	29,050	2254,394531
8	29,410	2162,109375
9	30,490	1902,832031
10	31,930	1792,96875
11	33,150	1687,5
12	32,720	1582,03125
13	31,820	1595,214844

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
Promedio	30,137	2099,233774

Tabla 6. Dirección Este, velocidad de vehículo 60km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	53,850	2992,675781
2	53,380	3247,558594
3	51,910	3458,496094
4	51,730	3594,726563
5	51,800	3651,855469
6	53,240	3471,679688
7	54,750	3177,246094
8	53,490	2961,914063
9	54,070	2852,050781
10	56,620	2786,132813
11	55,800	2900,390625
12	54,970	3049,804688
13	54,210	3317,871094
14	53,380	3678,222656

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
15	51,260	3854,003906
16	48,130	3884,765625
17	42,220	3744,140625
Promedio	52,636	3330,796186

Tabla 7. Dirección Oeste, velocidad de vehículo 20km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	21,670	905,2734375
2	22,570	900,8789062
3	22,170	936,0351562
4	21,600	966,796875
5	19,830	979,9804687
6	17,560	953,6132812
7	17,560	1006,347656
8	17,380	1015,136719
9	17,560	1037,109375
10	17,780	1037,109375
11	19,400	1041,503906

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
12	19,580	1076,660156
13	20,840	1063,476563
14	23,250	1045,898438
15	21,920	1089,84375
16	21,240	1050,292969
17	21,850	1067,871094
Promedio	20,221	1010,225184

Tabla 8. Dirección Oeste, velocidad de vehículo 40km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	31,710	1898,4375
2	32,790	1876,464844
3	32,500	1902,832031
4	35,560	1876,464844
5	34,840	1924,804688
6	35,710	1973,144531
7	34,230	2047,851563
8	32,500	2113,769531

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
9	34,300	2135,742188
10	36,500	2197,265625
11	38,910	2206,054688
12	35,350	2197,265625
13	34,300	2113,769531
14	35,710	2065,429688
15	34,410	2021,484375
16	35,380	1907,226563
17	30,600	1854,492188
Promedio	33,974	1819,335938

Tabla 9. Dirección Oeste, velocidad de vehículo 60km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	53,780	3282,714844
2	54,970	3234,375
3	57,200	3287,109375
4	55,800	3445,3125
5	57,740	3691,40625

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
6	54,180	3875,976563
7	53,960	3805,664063
8	54,900	3717,773438
9	57,200	3467,285156
10	60,190	3291,503906
11	57,160	3137,695313
12	58,710	3159,667969
13	55,470	3199,21875
Promedio	56,251	3430,438702

Tabla 10. Dirección Sur, velocidad de vehículo 20 km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	20,770	817,3828125
2	21,380	791,015625
3	20,620	852,5390625
4	18,680	874,5117187
5	16,630	949,21875
6	16,840	953,6132812

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
7	17,780	1010,742188
8	19,650	1019,53125
9	20,770	1063,476563
10	21,240	1107,421875
11	21,380	1160,15625
12	20,300	1133,789063
13	16,950	1116,210938
14	16,950	1204,101563
15	16,840	1107,421875
16	16,630	1168,945313
17	16,230	1125
Promedio	18,802	1026,769301

Tabla 11. Dirección Sur, velocidad de vehículo 40km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	34,090	2474,121094
2	35,100	2197,265625
3	35,130	1933,59375

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
4	34,990	1819,335938
5	34,050	1713,867188
6	34,300	1582,03125
7	34,050	1520,507813
8	34,050	1524,902344
9	34,560	1551,269531
10	35,850	1617,1875
11	38,160	1705,078125
12	37,760	1814,941406
13	38,590	1942,382813
14	38,410	2131,347656
Promedio	35,649	1823,41657

Tabla 12. Dirección Sur, velocidad de vehículo 60km/h

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
1	47,190	3111,328125
2	47,800	3634,277344
3	45,030	3928,710938

Muestra	Velocidad anemómetro comercial (km/h)	Velocidad anemómetro asimétrico (grad/s)
4	47,370	4091,308594
5	44,420	3884,765625
6	42,980	3418,945313
7	45,100	3045,410156
8	45,210	2772,949219
9	45,210	2742,1875
10	44,130	2865,234375
11	42,800	3265,136719
12	44,240	3779,296875
Promedio	45,123	3378,295898

7. APÉNDICE 3: MATRICES DE LOS SISTEMAS DE CLASIFICACIÓN

7.1. MATRICES DEL SISTEMA DE CLASIFICACIÓN 1

Tabla 13. Matriz de generadores (M) del sistema de clasificación 1.

Norte	Este	Oeste	Sur
0.065094455490411 9	0	0	0.236524958395633
0.130188910980824	0	0	0.118262479197817
0.130188910980824	0	0	0
0	0	0	0
0	0	0	0
0.195283366471236	0	0	0
0.260377821961648	0	0	0
0.260377821961648	0	0	0

Norte	Este	Oeste	Sur
0.455661188432884	0	0	0
0.065094455490411 9	0	0	0
0.195283366471236	0	0	0
0.325472277452060	0	0	0
0.195283366471236	0	0	0
0.065094455490411 9	0	0	0
0.130188910980824	0	0	0
0.130188910980824	0	0	0
0.130188910980824	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0.041344911529736 2	0
0	0	0.289414380708153	0
0	0	0.454794026827098	0
0	0	0.082689823059472 3	0
0	0	0.330759292237889	0
0	0	0.206724557648681	0
0	0	0.165379646118945	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0.159872153395486	0	0

Norte	Este	Oeste	Sur
0	0.119904115046615	0	0
0.065094455490411 9	0.159872153395486	0	0
0	0.359712345139844	0	0
0.065094455490411 9	0.559552536884202	0	0
0.065094455490411 9	0.039968038348871 6	0	0.059131239598908 3
0.065094455490411 9	0	0	0.059131239598908 3
0	0	0	0.354787437593450
0	0	0	0.295656197994541
0	0	0	0.177393718796725
0.065094455490411 9	0	0	0.295656197994541
0.065094455490411 9	0	0	0.059131239598908 3

Norte	Este	Oeste	Sur
0	0	0	0.118262479197817
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0.0650944554904119	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0.065094455490411 9	0	0	0
0.130188910980824	0	0	0
0.130188910980824	0	0	0
0.130188910980824	0	0	0
0.065094455490411 9	0	0	0
0.195283366471236	0	0	0
0.065094455490411 9	0	0	0
0.130188910980824	0	0	0
0.130188910980824	0	0	0

Norte	Este	Oeste	Sur
0.130188910980824	0	0	0
0.130188910980824	0	0	0
0.065094455490411 9	0	0	0
0.065094455490411 9	0	0	0
0.260377821961648	0	0	0
0.130188910980824	0	0	0
0	0	0.124034734589208	0
0.065094455490411 9	0	0.413449115297362	0
0.130188910980824	0	0.454794026827098	0
0	0	0.206724557648681	0
0	0	0.289414380708153	0

Norte	Este	Oeste	Sur
0.130188910980824	0	0.041344911529736 2	0
0	0	0	0
0.065094455490411 9	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0.279776268442101	0	0
0	0.279776268442101	0	0
0	0.319744306790973	0	0

Norte	Este	Oeste	Sur
0.0650944554904119	0.479616460186459	0	0
0	0.0399680383488716	0	0
0	0	0	0
0	0	0	0.591312395989083
0	0	0	0.473049916791266

Tabla 14. Matriz métrica (G) del sistema de clasificación 1.

Norte	Este	Oeste	Sur
1	0.0806526284935307	0.0915050331131326	0.0615858535066243
0.0806526284935307	1	0	0.0023633596 5190548
0.0915050331131326	0	1	0
0.0615858535066243	0.0023633596 5190548	0	1

La matriz de proyección D viene definida como [4][120], sin embargo, para representarla de una mejor manera en el espacio disponible se ha considerado transponerla, obteniendo D[120][4].

Tabla 15. Matriz de proyección (D) transpuesta del sistema de clasificación 1

Norte	Este	Oeste	Sur
0.0515338401130995	-0.00470785877466668	-0.00471560574599605	0.233362329231269
0.1252639659386667	-0.0103641906587328	-0.0114622833511001	0.110572485251887
0.1326627278428230	-0.0106803483618660	-0.0121393073041361	-0.0081449058183305600
0	0	0	0
0	0	0	0
0.1989940917642350	-0.0160205225427990	-0.0182089609562041	-0.0122173587274958
0.2653254556856470	-0.0213606967237320	-0.0242786146082721	-0.0162898116366611

Norte	Este	Oeste	Sur
0.2653254556856470	-0.0213606967237320	-0.0242786146082721	-0.0162898116366611
0.4643195474498820	-0.0373812192665310	-0.0424875755644762	-0.0285071703641570
0.0663313639214117	-0.0053401741809330000	-0.0060696536520680300	-0.0040724529091652828
0.1989940917642350	-0.0160205225427990	-0.0182089609562041	-0.0122173587274958
0.3316568196070590	-0.0267008709046650	-0.0303482682603401	-0.0203622645458264
0.1989940917642350	-0.0160205225427990	-0.0182089609562041	-0.0122173587274958
0.0663313639214117	-0.0053401741809330000	-0.0060696536520680300	-0.0040724529091652828

Norte	Este	Oeste	Sur
0.13266272784282 3	-0.0106803483618660	-0.0121393073041361	-0.00814490581833056
0.13266272784282 3	-0.0106803483618660	-0.0121393073041361	-0.00814490581833056
0.13266272784282 3	-0.0106803483618660	-0.0121393073041361	-0.00814490581833056
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
-0.003855156193722307	0.000310369097695251	0.041697677724899	0.000236689570789093
-0.026986093356061 5	0.002172583683866 76	0.291883744074294	0.001656826995523 65
-0.042406718130953 8	0.003414060074647 76	0.458674454973890	0.002603585278680 03

Norte	Este	Oeste	Sur
-0.00771031238744615	0.000620738195390502	0.0833953554497982	0.000473379141578187
-0.03084124954978460	0.002482952781562010	0.333581421799193	0.001893516566312750
-0.01927578096861540	0.001551845488476260	0.208488388624495	0.001183447853945470
-0.01542062477489230	0.001241476390781000	0.166790710899596	0.000946758283156373
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
-0.01311548179304590	0.160928941387546	0.001200132595767350	0.000427395173476135
-0.00983661134478443	0.120696706040660	0.000900099446825515	0.000320546380107102

Norte	Este	Oeste	Sur
0.0532158821283658	0.155588767206613	-0.00486952105630067	-0.00364505773568915
-0.0295098340343533	0.362090118121979	0.00270029834047655	0.000961639140321305
0.0204271776457510	0.557911120675478	-0.00186918956688229	-0.00257656980199881
0.0593531125210722	0.035050140017520 1	-0.00543110852660819	0.055393091419312 3
0.0626319829693336	-0.00518209532936642	-0.00573114167555003	0.055286242625943 3
-0.0221962857124684	0.000948473109399470	0.00203107185910797	0.356152173210651
-0.0184969047603903	0.000790394257832892	0.00169255988258998	0.296793477675543

Norte	Este	Oeste	Sur
-0.011098142856234	0.000474236554699735	0.001015535929553 99	0.178076086605326
0.047834459161021 4	-0.004549779923100 10	-0.004377093769478 05	0.292721024766377
0.062631982969333 6	-0.005182095329366 42	-0.005731141675550 03	0.055286242625943 3
-0.007398761904156 13	0.000316157703133157	0.000677023953035990	0.118717391070217
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0.0663313639214117	-0.00534017418093300	-0.00606965365206803	-0.00407245290916528
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0.0663313639214117	-0.00534017418093300	-0.00606965365206803	-0.00407245290916528
0.1326627278428230	-0.01068034836186601	-0.01213930730413611	-0.00814490581833056
0.1326627278428230	-0.01068034836186601	-0.01213930730413611	-0.00814490581833056

Norte	Este	Oeste	Sur
0.132662727842823	-0.0106803483618660	-0.0121393073041361	-0.00814490581833056
0.066331363921411 7	-0.0053401741809330 0	-0.0060696536520680 3	-0.0040724529091652 8
0.198994091764235	-0.0160205225427990	-0.0182089609562041	-0.0122173587274958
0.066331363921411 7	-0.0053401741809330 0	-0.0060696536520680 3	-0.0040724529091652 8
0.132662727842823	-0.0106803483618660	-0.0121393073041361	-0.00814490581833056
0.132662727842823	-0.0106803483618660	-0.0121393073041361	-0.00814490581833056
0.132662727842823	-0.0106803483618660	-0.0121393073041361	-0.00814490581833056

Norte	Este	Oeste	Sur
0.132662727842823	-0.0106803483618660	-0.0121393073041361	-0.00814490581833056
0.066331363921411 7	-0.0053401741809330 0	-0.0060696536520680 3	-0.0040724529091652 8
0.066331363921411 7	-0.0053401741809330 0	-0.0060696536520680 3	-0.0040724529091652 8
0.265325455685647	-0.0213606967237320	-0.0242786146082721	-0.0162898116366611
0.132662727842823	-0.0106803483618660	-0.0121393073041361	-0.00814490581833056
-0.011565468581169 2	0.000931107293085753	0.125093033174697	0.000710068712367280
0.027779801984181 0	-0.0022364832039804 9	0.410907123596923	-0.0017055572012743 5

Norte	Este	Oeste	Sur
0.0902560097118696	-0.00726628828721823	0.446535147669754	-0.00554132053965054
-0.019275780968615 4	0.001551845488476 26	0.208488388624495	0.001183447853945 47
-0.026986093356061 5	0.002172583683866 76	0.291883744074294	0.001656826995523 65
0.128807571649100	-0.01036997926417 07	0.029558370420763 0	-0.00790821624754147
0	0	0	0
0.066331363921411 7	-0.00534017418093300	-0.00606965365206803	-0.00407245290916528
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
-0.022952093137830 3	0.281625647428206	0.002100232042592 87	0.000747941553583237
-0.022952093137830 3	0.281625647428206	0.002100232042592 87	0.000747941553583237
-0.026230963586091 8	0.321857882775092	0.002400265191534 71	0.000854790346952271
0.026984918542274 0	0.477446649981705	-0.00246925586476597	-0.00279026738873688
-0.003278870448261 48	0.040232235346886 5	0.000300033148941838	0.000106848793369034
0	0	0	0
-0.036993809520780 6	0.001580788515665 78	0.003385119765179 95	0.593586955351085

Norte	Este	Oeste	Sur
-0.029595047616624 5	0.001264630812532 63	0.002708095812143 96	0.474869564280868

7.2. MATRICES DEL SISTEMA DE CLASIFICACIÓN 2

Tabla 16. Matriz de generadores (M) del sistema de clasificación 2.

Norte	Este	Oeste	Sur
0.034482758620689 7	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0.060412209330177	0	0
0	0.060412209330177	0	0
0	0.604122093330177	0	0
0	0.362473255998106	0	0
0	0.181236627999053	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0.119522860933439
0	0	0	0.478091443733757
0	0	0	0.298807152333598
0	0	0	0.239045721866879
0	0	0	0.179284291400159
0.0689655172413793	0	0	0.119522860933439

Norte	Este	Oeste	Sur
0.241379310344828	0	0	0
0.310344827586207	0	0	0
0.413793103448276	0	0	0
0.310344827586207	0	0	0
0.103448275862069	0	0.211079263419088	0
0	0	0.281439017892117	0
0.034482758620689 7	0	0.281439017892117	0
0	0	0.281439017892117	0
0	0.060412209333017 7	0.351798772365146	0
0	0	0.281439017892117	0
0	0	0.140719508946058	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0.0703597544730292	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0.241648837332071	0	0
0	0.483297674664142	0	0
0	0.362473255998106	0	0
0	0.181236627999053	0	0
0	0.060412209330177	0	0
0	0	0	0
0	0	0.070359754473029 2	0

Norte	Este	Oeste	Sur
0	0	0.0703597544730292	0
0	0	0.0703597544730292	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0.0703597544730292	0
0	0	0.0703597544730292	0
0	0	0.0703597544730292	0
0	0	0.281439017892117	0.0597614304667197

Norte	Este	Oeste	Sur
0	0	0.211079263419088	0.298807152333598
0	0	0.070359754473029 2	0.478091443733757
0	0	0.351798772365146	0.119522860933439
0.172413793103448	0	0.351798772365146	0.478091443733757
0.482758620689655	0	0.281439017892117	0
0.310344827586207	0	0.070359754473029 2	0
0.448275862068966	0	0	0
0	0	0	0

Tabla 17. Matriz métrica (G) del sistema de clasificación 2.

Norte	Este	Oeste	Sur
1	0	0.249898438300759	0.090672515190885 0
0	1	0.021252941079221 8	0

Norte	Este	Oeste	Sur
0.249898438300759	0.021252941079221 8	1	0.323769567243846
0.090672515190885 0	0	0.323769567243846	1

Igual como en el sistema anterior, se procede a coloca la matriz de proyección (D) transpuesta.

Tabla 18. Matriz de proyección (D) transpuesta del sistema de clasificación 2.

Norte	Este	Oeste	Sur
0.036784871271759 7	0.000192703927959904	-0.009067165209821 40	-0.000399704643071054
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0.000337608431021046	0.060444476200715 0	-0.001518230704026 33	0.000460945092428599
0.000337608431021046	0.060444476200715 0	-0.001518230704026 33	0.000460945092428599
0.003376084310210 46	0.604444762007150	-0.015182307040263 3	0.004609450924285 99
0.002025650586126 28	0.362666857204290	-0.009109384224158 01	0.002765670554571 60
0.001012825293063 14	0.181333428602145	-0.004554692112079 00	0.001382835277285 80
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
-0.00138544143157872	0.000911959300753201	-0.0429097929248384	0.133541367478488
-0.00554176572631488	0.003647837203012 80	-0.1716391716993 54	0.534165469913953

Norte	Este	Oeste	Sur
-0.0034636035789468	0.00227989825188300	-0.107274482312096	0.333853418696221
-0.0027708828631574	0.00182391860150640	-0.085819585849676	0.267082734956977
-0.0020781621473680	0.00136793895112980	-0.064364689387257	0.200312051217732
0.072184301119407	0.00129736715667301	-0.061044123344481	0.132741958192346
0.257494098902318	0.00134892749571933	-0.063470156468749	-0.002797932501497
0.331063841445838	0.00173433535163914	-0.081604486888392	-0.003597341787639
0.441418455261117	0.00231244713551885	-0.108805982517857	-0.004796455716852

Norte	Este	Oeste	Sur
0.331063841445838	0.00173433535163914	-0.0816044868883926	-0.0035973417876394 9
0.054851787755419 1	-0.0047265612655625 2	0.22239563211246 5	-0.0769784871244749
-0.0740037680798134	-0.0070728973992563 1	0.33279617032257 2	-0.1010391642603 49
-0.0372188968080536	-0.0068801934712964 0	0.32372900511275 1	-0.1014388689034 20
-0.0740037680798134	-0.0070728973992563 1	0.33279617032257 2	-0.1010391642603 49
-0.0921671016687457	0.05160335445164 47	0.41447698219918 9	-0.1258380102330 08
-0.0740037680798134	-0.0070728973992563 1	0.33279617032257 2	-0.1010391642603 49

Norte	Este	Oeste	Sur
-0.0370018840399067	-0.00353644869962815	0.166398085161286	-0.0505195821301745
0	0	0	0
0	0	0	0
0	0	0	0
-0.0185009420199533	-0.00176822434981408	0.0831990425806431	-0.0252597910650872
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0.00135043372408419	0.241777904802860	-0.00607292281610534	0.00184378036971440
0.00270086744816837	0.483555809605720	-0.01214584563221 07	0.00368756073942879
0.00202565058612628	0.362666857204290	-0.00910938422415801	0.00276567055457160
0.00101282529306314	0.181333428602145	-0.00455469211207900	0.00138283527728580

Norte	Este	Oeste	Sur
0.000337608431021046	0.060444476200715	-0.0015182307040263	0.000460945092428599
0	0	0	0
-0.0185009420199533	-0.0017682243498140	0.0831990425806431	-0.0252597910650872
-0.0185009420199533	-0.0017682243498140	0.0831990425806431	-0.0252597910650872
-0.0185009420199533	-0.0017682243498140	0.0831990425806431	-0.0252597910650872
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Norte	Este	Oeste	Sur
-0.0185009420199533	-0.0017682243498140 8	0.0831990425806431	-0.0252597910650872
-0.0185009420199533	-0.0017682243498140 8	0.0831990425806431	-0.0252597910650872
-0.0185009420199533	-0.0017682243498140 8	0.0831990425806431	-0.0252597910650872
-0.0746964887956027	-0.0066169177488797 1	0.3113412738601 53	-0.0342684805211048
-0.0589664296388068	-0.0030247747975592 3	0.1423226454298 33	0.2580740455009 59
-0.0240427077462682	0.0018796128531987 3	-0.0884401291187105	0.5089056788488 66
-0.0938901515313454	-0.0079291624483171 8	0.3730854199783 77	0.0072424121530 5205

Norte	Este	Oeste	Sur
0.085877880532717 1	-0.004229764906258 06	0.19902021515475 5	0.405867991373162
0.440984429724823	-0.004375042407817 65	0.20585585738507 3	-0.106635029263344
0.312562899425884	-3,39E+09	0.00159455569225052	-0.028857132852726 7
0.478203326532876	0.00250515106347876	-0.117873147727678	-0.005196160359923 70
0	0	0	0

8. APÉNDICE 4: GRÁFICAS DE VALIDACIÓN DE LOS SISTEMAS DE CLASIFICACIÓN

8.1. GRÁFICAS DEL SISTEMA DE CLASIFICACIÓN 1

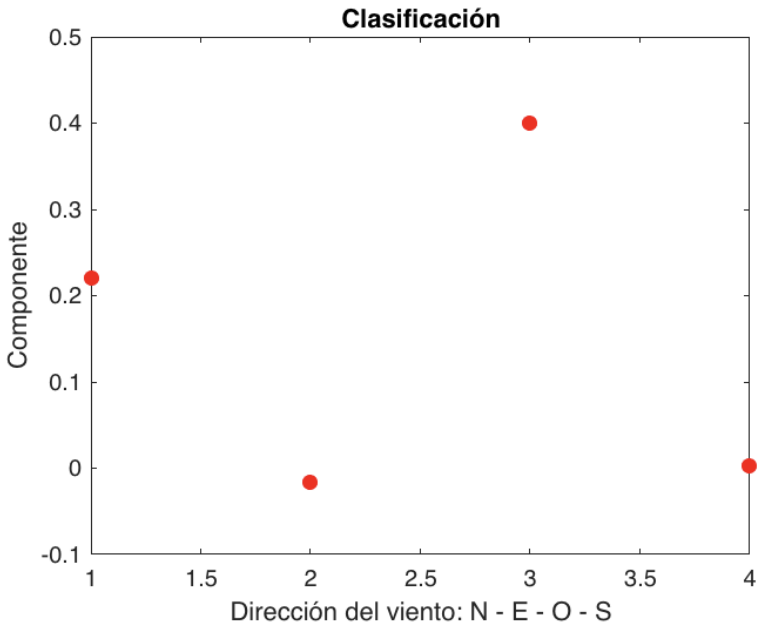


Ilustración 27. Clasificación de dirección Oeste, velocidad 60km/h.

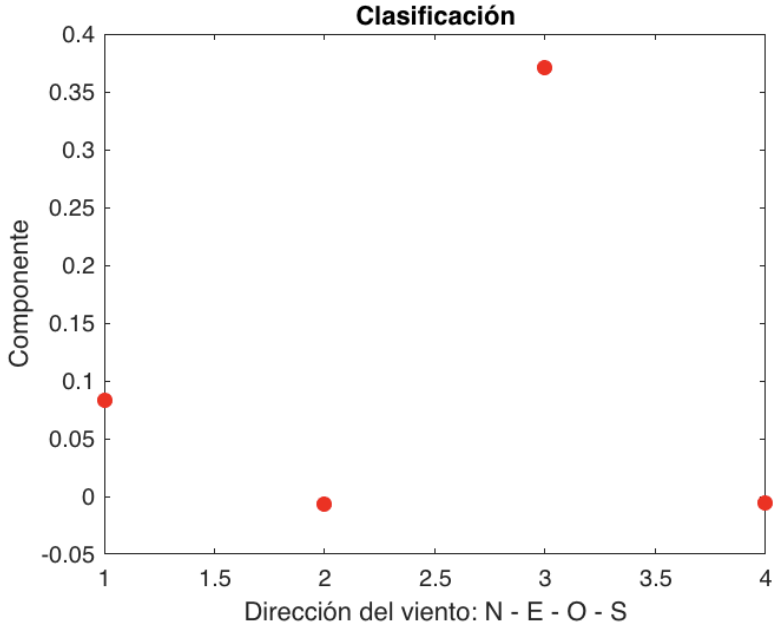


Ilustración 28. Clasificación de dirección Oeste, velocidad 20km/h.

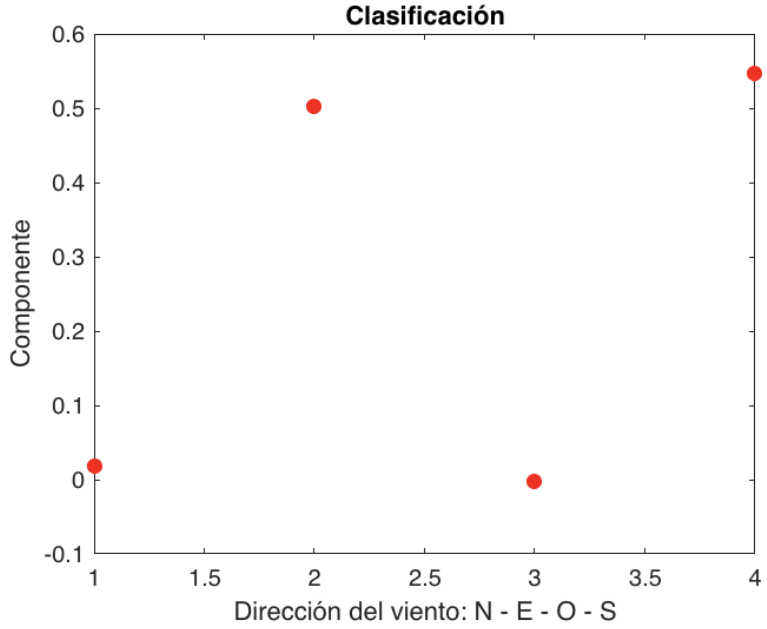


Ilustración 29. Clasificación de dirección Sur, velocidad 60km/h.

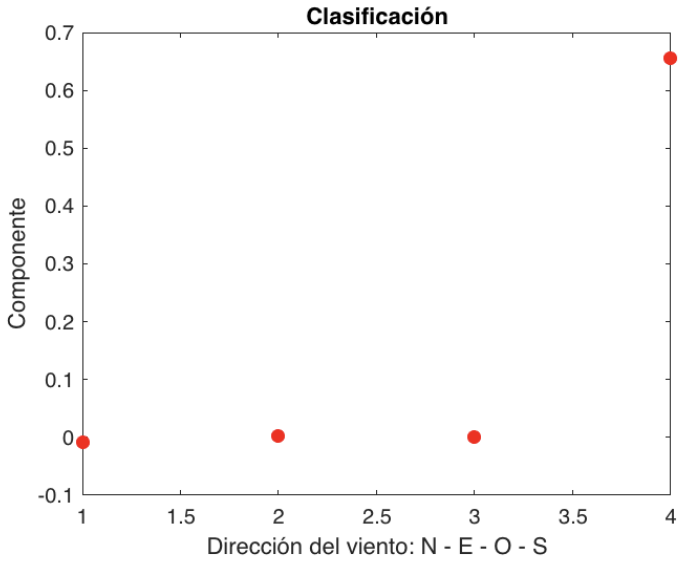


Ilustración 30. Clasificación de dirección Sur, velocidad 20km/h.

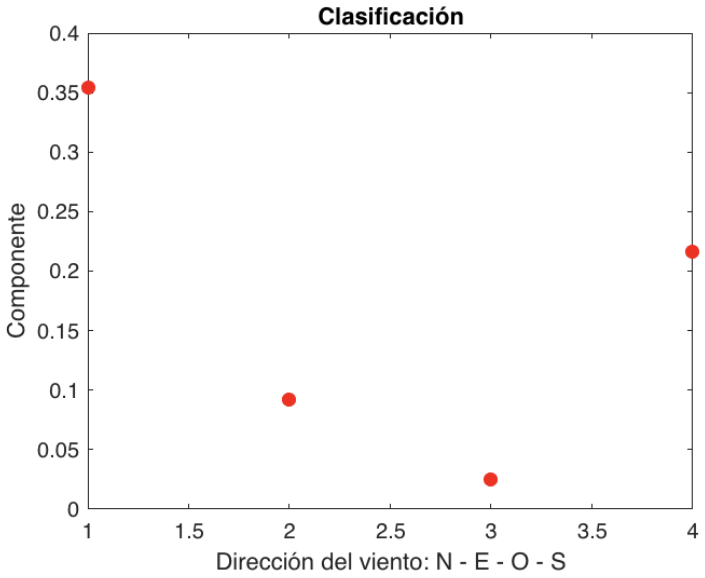


Ilustración 31. Clasificación de la dirección Norte, velocidad 60km/h.

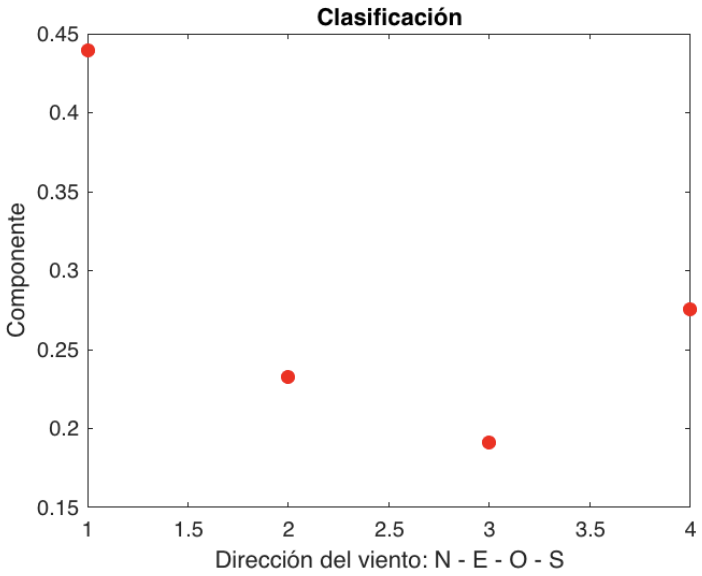


Ilustración 32. Clasificación de la dirección Norte, velocidad 20km/h.

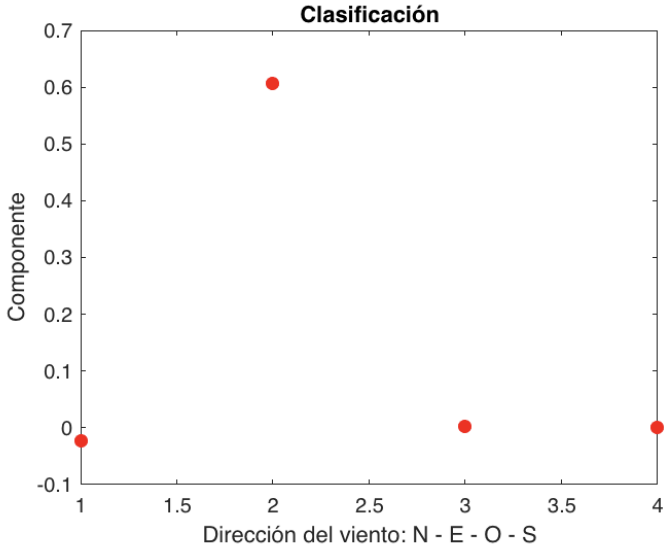


Ilustración 33. Clasificación de dirección Este, velocidad 60km/h.

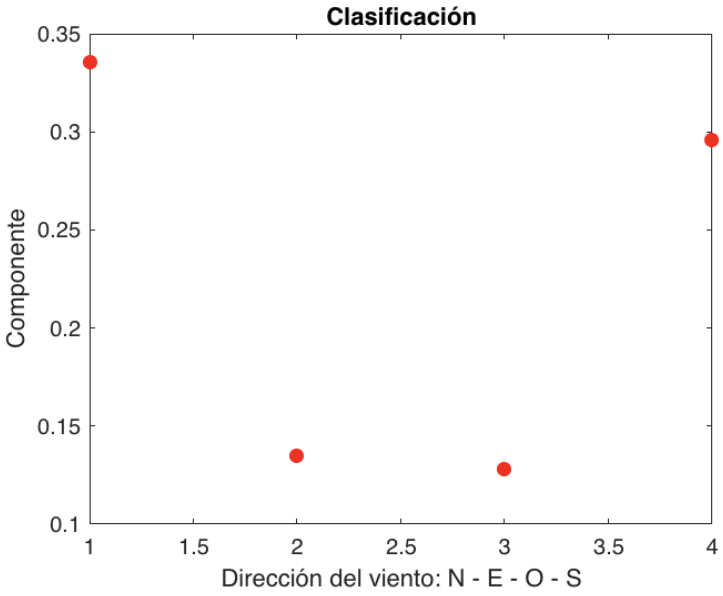


Ilustración 34.Clasificación de dirección Este, velocidad 20km/h.

8.2. GRÁFICAS DEL SISTEMA DE CLASIFICACIÓN 2

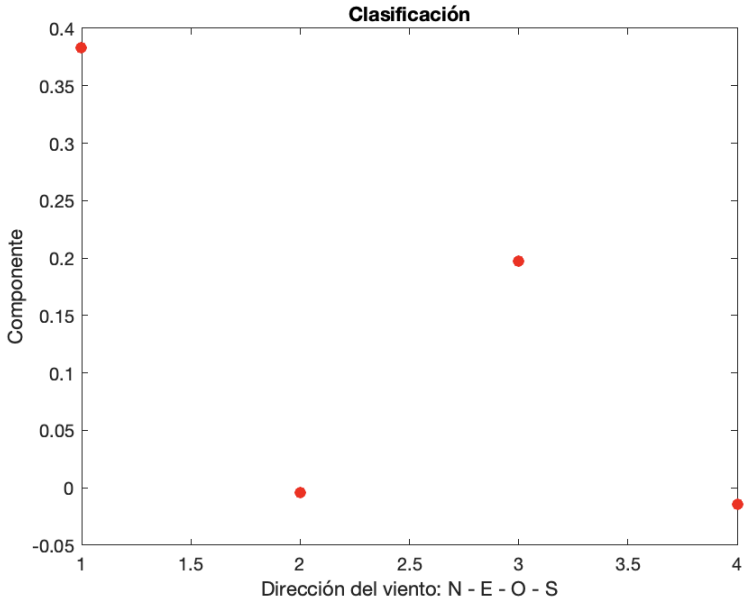


Ilustración 35. Clasificación de dirección Oeste, velocidad 60km/h.

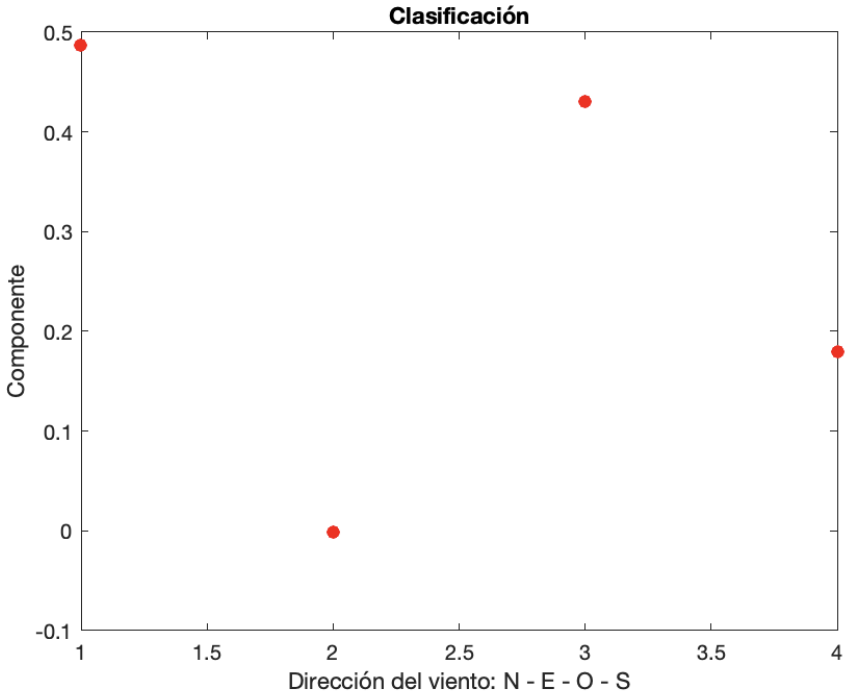


Ilustración 36. Clasificación de dirección Oeste, velocidad 40km/h.

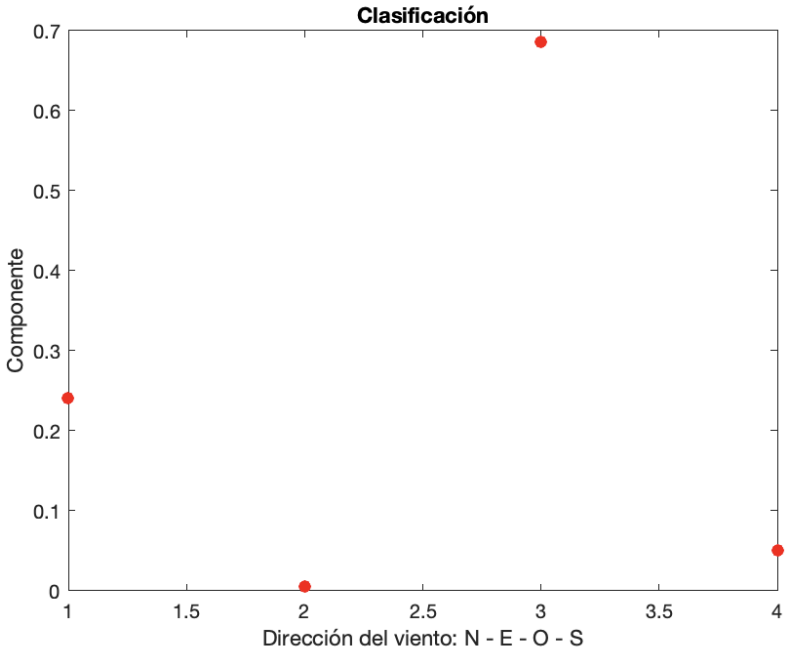


Ilustración 37. Clasificación dirección Oeste, velocidad 20km/h (v1).

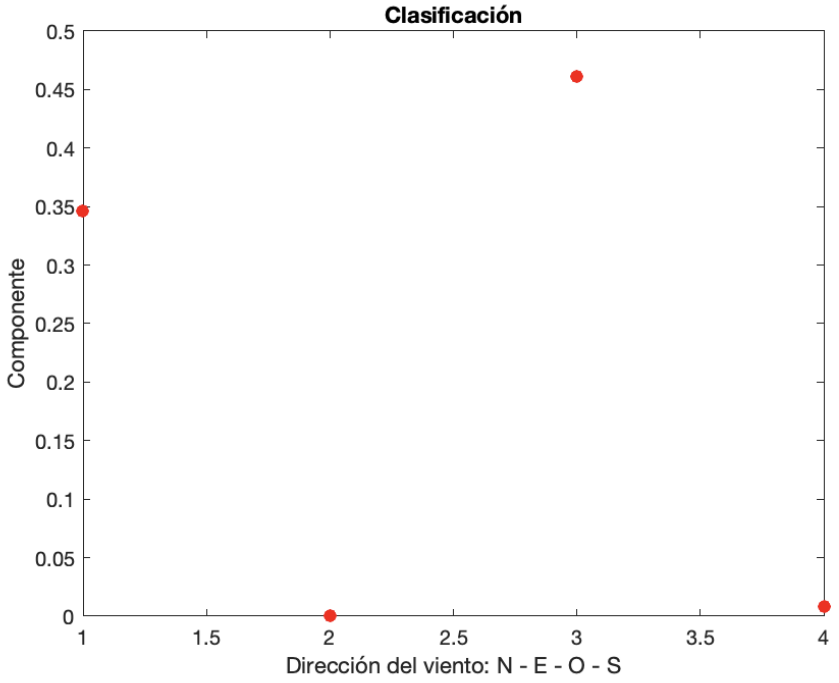


Ilustración 38. Clasificación dirección Oeste, velocidad 20km/h (v2).

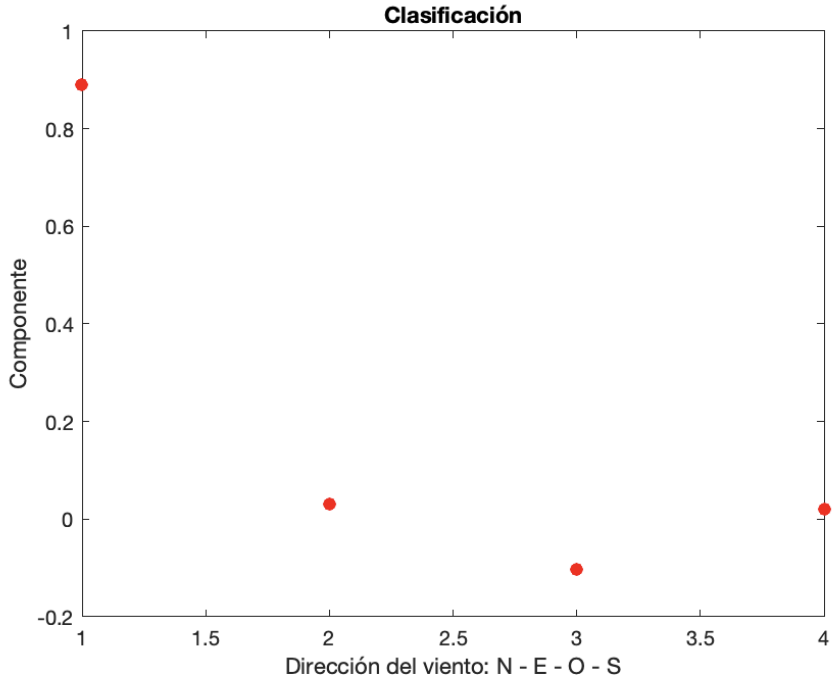


Ilustración 39. Clasificación de la dirección Sur, velocidad 20 km/h (v1)

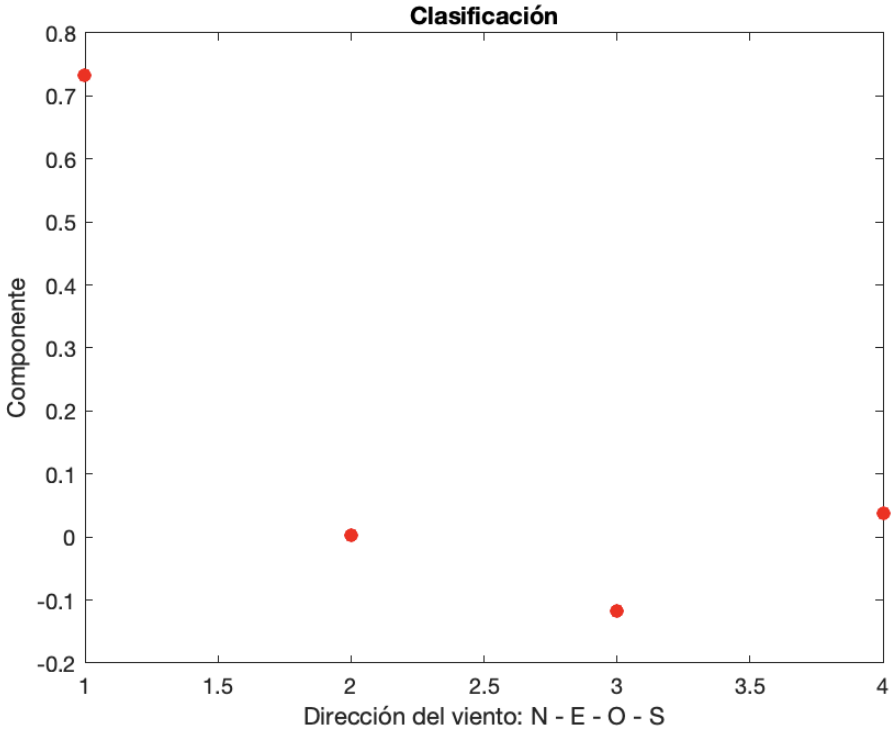


Ilustración 40. Clasificación dirección Sur, velocidad 20km/h (v2).

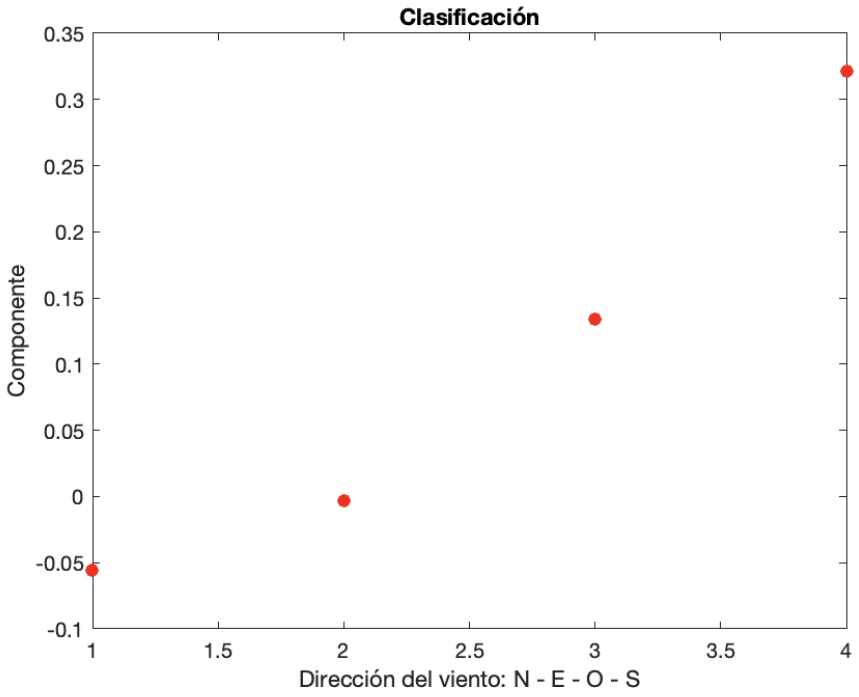


Ilustración 41. Clasificación de la dirección Sur, velocidad 40km/h.

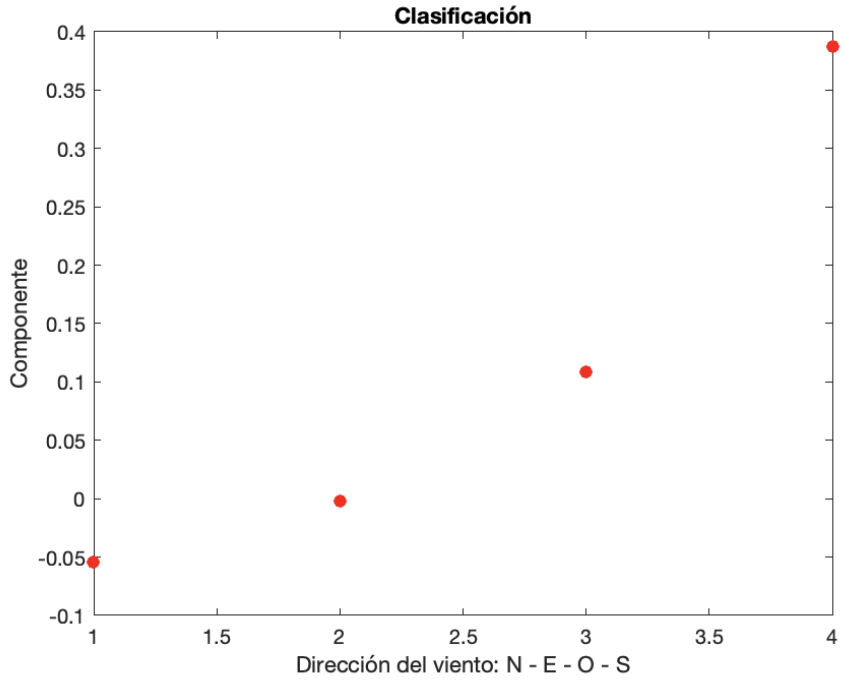


Ilustración 42. Clasificación dirección Sur, velocidad 60km/h. (v1)

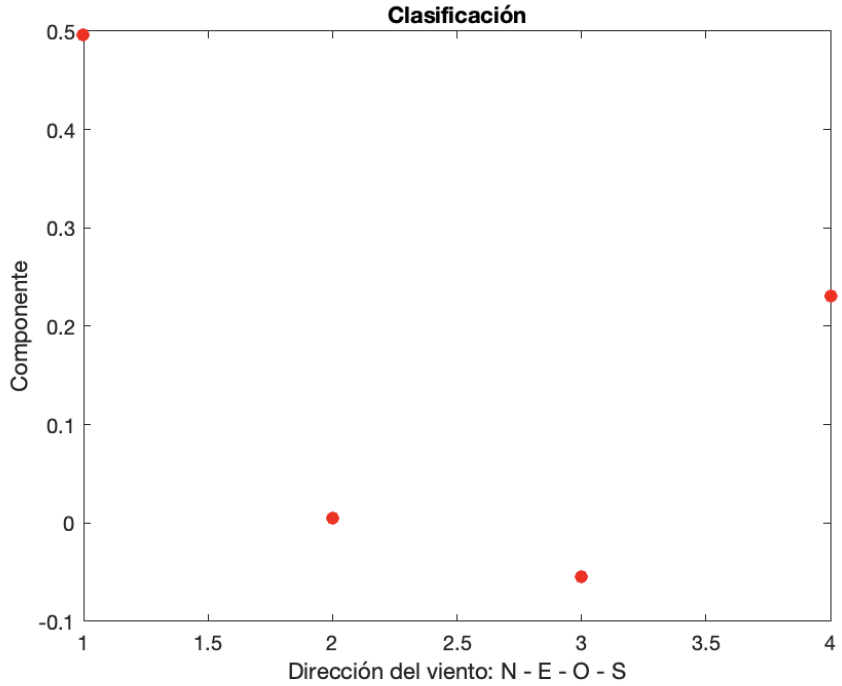


Ilustración 43. Clasificación dirección Sur, velocidad 60km/h (v2).

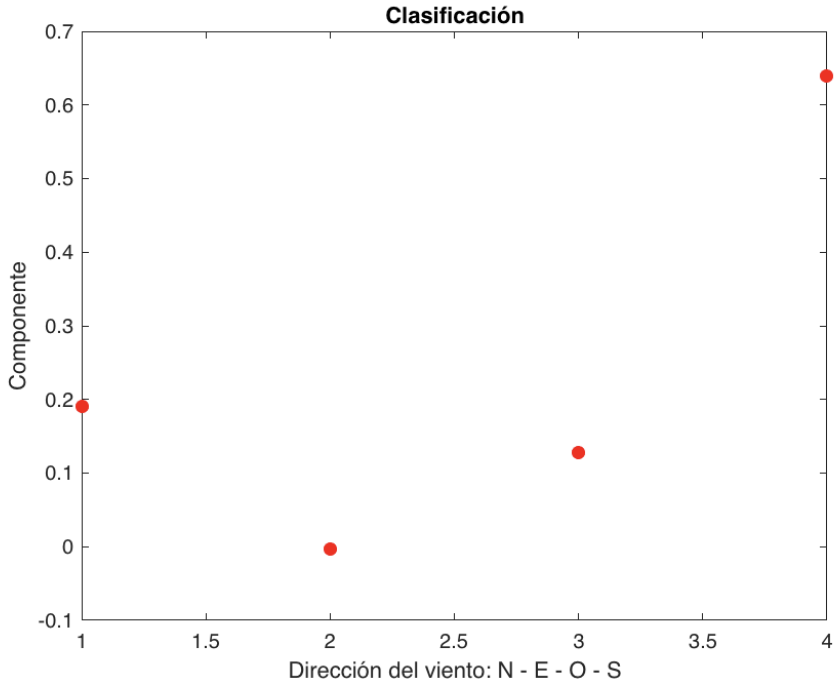


Ilustración 44. Clasificación de la dirección Norte, velocidad 60km/h (v1).

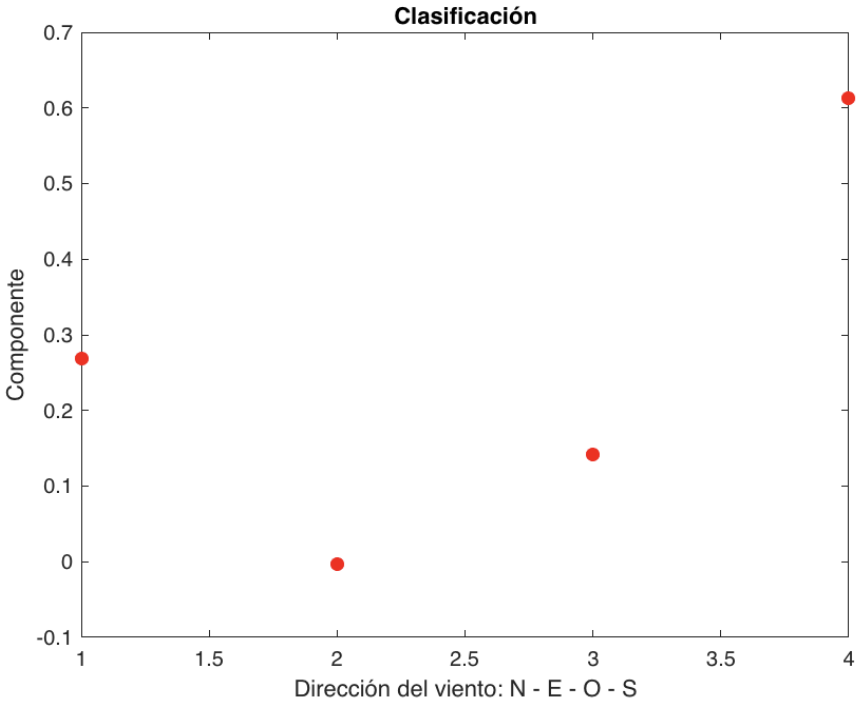


Ilustración 45. Clasificación de la dirección Norte, velocidad 60km/h (v2).

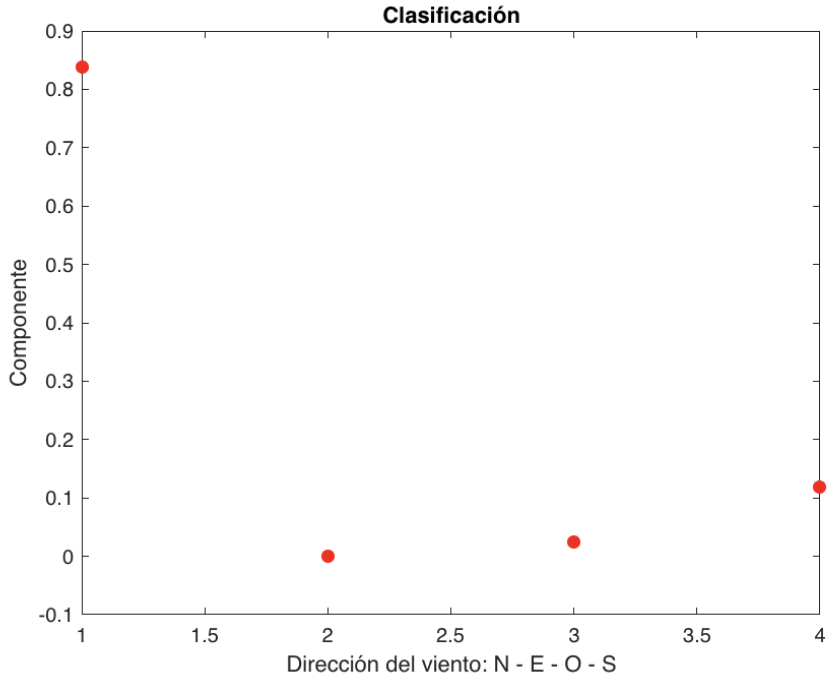


Ilustración 46. Clasificación dirección Norte, velocidad 40km/h.

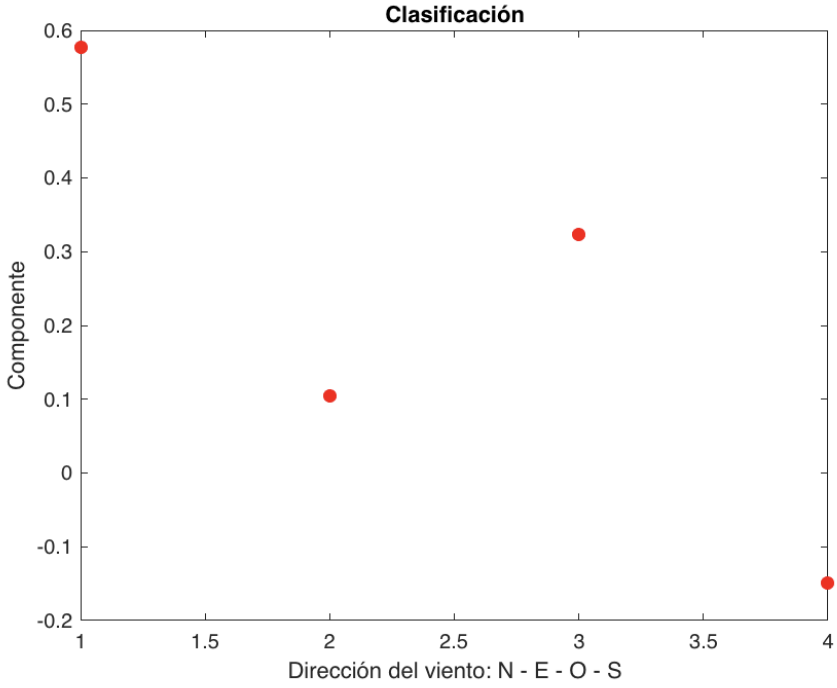


Ilustración 47. Clasificación dirección Norte, velocidad 20km/h (v1).

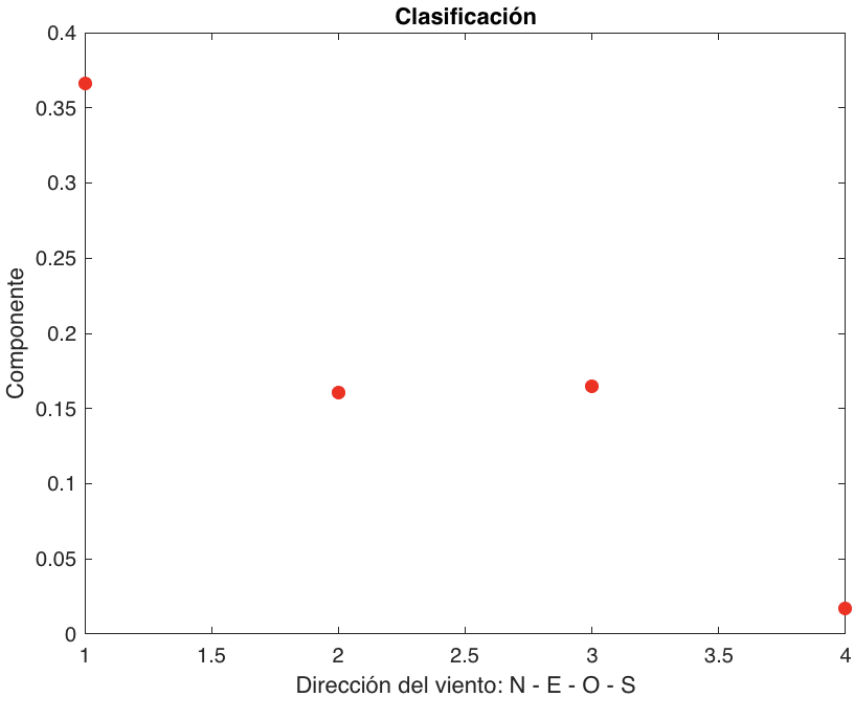


Ilustración 48. Clasificación de la dirección Norte, velocidad 20km/h (v2).

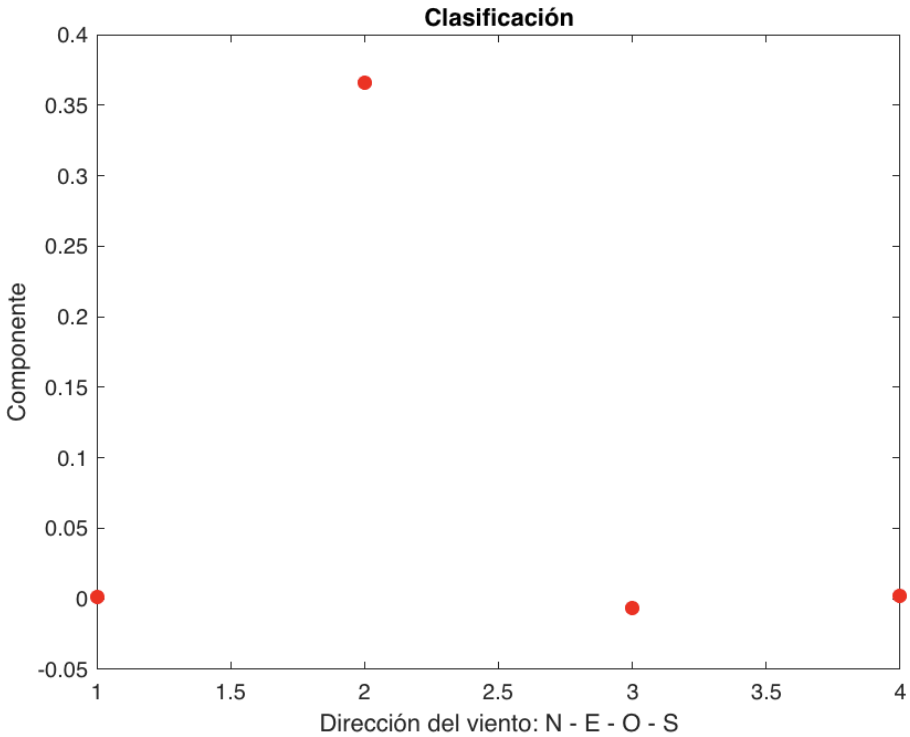


Ilustración 49. Clasificación de la dirección Este, velocidad 40km/h.

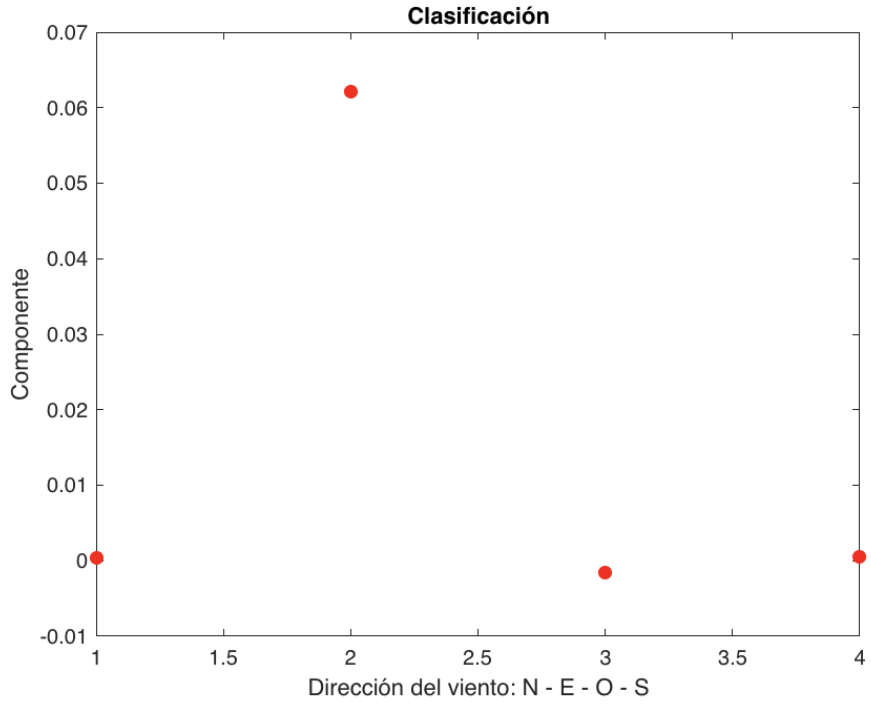


Ilustración 50. Clasificación de la dirección Este, velocidad 60km/h.

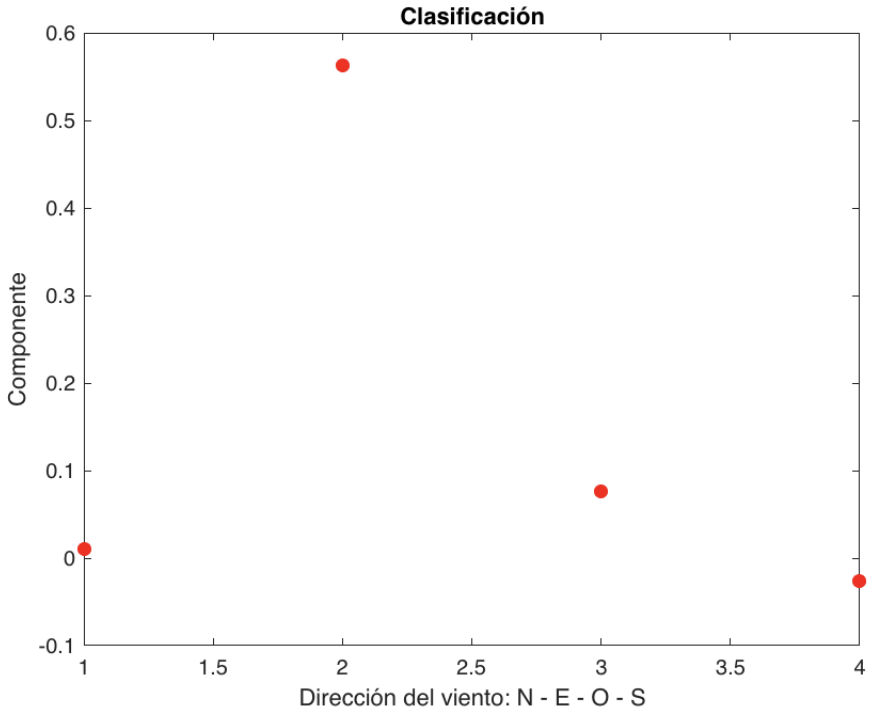
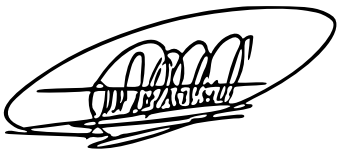


Ilustración 51. Clasificación dirección Este, velocidad 20km/h.

Relación de documentos

(_) Memoria 185 páginas
(X) Anexos 126 páginas

La Almunia, a 23 de 11 de 2022



Firmado: Jonathan Chamba Benítez