



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Influencia de las variables meteorológicas en las simulaciones
térmicas de buques. El Patrullero Tabarca como caso de estudio.*

Grado en Ingeniería Mecánica

ALUMNO: Pablo Dodero Vázquez

DIRECTORES: M^a Elena Arce Fariña

Andrés Suárez García

CURSO ACADÉMICO: 2015-2016

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Influencia de las variables meteorológicas en las simulaciones
térmicas de buques. El Patrullero Tabarca como caso de estudio.*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

RESUMEN

Las simulaciones térmicas están cobrando una gran importancia durante los últimos años en el ámbito de la construcción en busca del ahorro energético y la eficiencia térmica. Es por ello que resulta necesario realizar simulaciones calibradas. En este sentido, el estudio de la influencia de las variables meteorológicas resulta clave.

El objetivo principal de este Trabajo de Fin de Grado es analizar la influencia de variables meteorológicas en simulaciones térmicas aplicadas a buques. Para el análisis de la influencia e importancia de considerar las variables meteorológicas se empleó el software de simulación Trnsys (TRaNsient System Simulation), programa que permite resolver sistemas energéticos complejos en edificaciones.

El análisis se hizo aplicado a un patrullero de la Armada Española (Tabarca). Para ello, además de utilizar las dimensiones del buque y geolocalizar la simulación su puerto base, se midieron las temperaturas en el interior. Además, se instaló una estación meteorológica para la toma medidas in situ. Se obtuvieron también datos meteorológicos de diferentes fuentes (Meteonorm y MeteoGalicia) para analizar y comparar los resultados.

Las conclusiones y líneas futuras de este trabajo se basan en la validación del modelo para el análisis, la ampliación del periodo analizado y la correcta ubicación de la estación meteorológica.

PALABRAS CLAVE

Simulación térmica en buques, variables meteorológicas, Trnsys, patrullero Tabarca, ubicación, estación meteorológica.

AGRADECIMIENTOS

Agradezco en primer lugar a mis tutores, Dra. M^a Elena Arce Fariña y Andrés Suárez García, por su constante implicación en el trabajo, disponibilidad para solucionar dudas y por saber orientarme para realizar este trabajo durante los dos meses de continuo trabajo y esfuerzo.

También me gustaría agradecer al equipo de investigación GTE de la Universidad de Vigo su ayuda facilitándonos datos para la realización de simulaciones en este trabajo.

Agradezco también a la dotación del Patrullero Tabarca por permitirme realizar todo el trabajo necesario a bordo y por la información facilitada.

Por último, me gustaría agradecer al personal de la Escuela Naval Militar y a todos los profesores del Centro Universitario de la Defensa que me han ayudado, por dedicar tiempo para que este trabajo llegue a buen puerto.

CONTENIDO

Contenido	1
Índice de Figuras	3
Índice de Tablas.....	8
Índice de Ecuaciones	9
1 Objetivos	10
2 Introducción.....	12
2.1 La simulación térmica de edificios	12
2.2 Importancia de las variables meteorológicas en la simulación térmica	12
2.3 Obtención de datos meteorológicos	13
2.3.1 Red de estaciones de MeteoGalicia	16
3 Estado del arte	20
3.1 Trabajos existentes sobre simulación térmica	20
3.2 Trabajos existentes sobre simulación térmica en buques.....	20
3.3 Software utilizado en la simulación térmica	21
4 Metodología.....	22
4.1 Software empleado.....	22
4.1.1 SketchUp.....	22
4.1.2 Trnsys	26
4.1.3 Meteonorm.....	33
4.1.4 Energy plus	35
4.2 Estación meteorológica usada	36
4.2.1 Descripción de los elementos y montaje	36
4.2.2 Montaje del sistema	42
4.2.3 Código usado	53
4.3 MeteoGalicia: datos temperatura del agua de mar.....	53
4.4 Construcción de archivos meteorológicos (.epw)	55
4.4.1 Variables utilizadas.....	55
4.4.2 Esquema de procedimiento de construcción de archivos	57
4.4.3 Datos meteorológicos a emplear.....	59
4.5 Barco objeto de análisis: Patrullero Tabarca.....	59
4.5.1 Descripción general del buque.....	59
4.5.2 Puerto base.....	61
4.5.3 Aspectos constructivos del barco. Cerramientos	61

4.6 Selección de variables objeto de análisis	64
4.6.1 Análisis de sensibilidad	64
4.6.2 Medición de temperatura interna del buque	82
4.6.3 Medición de la temperatura de agua de mar	89
5 Simulación térmica a bordo del patrullero Tabarca	95
5.1 Descripción de la simulación	95
5.2 Resultados de las simulaciones	102
6 Conclusiones y líneas futuras	106
6.1 Buque	106
6.2 Datos meteorológicos.....	107
6.3 Estación meteorológica	108
6.4 Simulación en Trnsys.....	109
7 Líneas Futuras	111
7.1 Validación del modelo y simulación en <i>Trnsys</i>	111
7.2 Datos meteorológicos y estación.....	111
8 Bibliografía.....	112
Anexo I: Escala de Beaufort.....	1
Anexo II: Planos patrullero Tabarca.....	2
Anexo III: Curvas Hidrostáticas patrullero Tabarca	6
Anexo IV: Conversión de día y mes a hora.....	7
Anexo V: Código Arduino	1
Anexo VI: Código Device Imp IDE	19
Anexo VII: Código Agent	33

ÍNDICE DE FIGURAS

Figura 2-1: Satélite meteorológico[5]	14
Figura 2-2: Radar Doppler [5].....	14
Figura 2-3: Estación meteorológica. Garita Stevenson [5]	15
Figura 2-4: Globo meteorológico [5]	15
Figura 2-5: Red de MeteoGalicia [6].....	16
Figura 2-6: Distancia entre patrullero Tabarca y estación de Lourizán [8].....	16
Figura 2-7: Estación meteorológica Lourizán [9]	17
Figura 2-8: Anemómetro [6]	18
Figura 2-9: Piranómetro [6].....	18
Figura 2-10: Pluviómetro [6].....	18
Figura 2-11: Sonda horas de sol [6]	18
Figura 2-12: Barómetro [6]	19
Figura 2-13: Sonda de temperatura y humedad [6]	19
Figura 4-1: <i>SketchUp</i> con plug-in <i>Trnsys</i>	22
Figura 4-2: Barra de Herramientas. Crear y guardar archivo	23
Figura 4-3: Barra de Herramientas. Crear zona térmica.....	23
Figura 4-4: Barra de Herramientas. Crear superficie	23
Figura 4-5: Barra de Herramientas. Extruir superficie	23
Figura 4-6: Barra de Herramientas. Información	23
Figura 4-7: Información del objeto.....	24
Figura 4-8: Esquema de zonas térmicas	24
Figura 4-9: Crear una zona térmica en <i>SketchUp</i>	25
Figura 4-10: Crear superficie en <i>SketchUp</i>	25
Figura 4-11: Extruir superficie para crear un bloque	26
Figura 4-12: Asignar tipo de pared en <i>SketchUp</i>	26
Figura 4-13: <i>Simulation Studio</i> de <i>Trnsys</i>	27
Figura 4-14: <i>Types Simulation Studio</i>	28
Figura 4-15: Selección del periodo de simulación	28
Figura 4-16: <i>TRNBuild</i>	29
Figura 4-17: Configuración de la zona en <i>TRNBuild</i>	29
Figura 4-18: Type 15-2: Datos meteorológicos	30
Figura 4-19: Archivo externo con datos meteorológicos.	30
Figura 4-20: Type 9b: Temperatura del agua de mar	31
Figura 4-21: Frecuencia de lectura de datos del type 9b	31

Figura 4-22: Type 56: <i>Building</i>	31
Figura 4-23: <i>External File</i> del type 56	32
Figura 4-24: Type 65c: <i>Online Plotter</i>	32
Figura 4-25: <i>Inputs</i> del type 65c.....	32
Figura 4-26: Conversor de unidades	33
Figura 4-27: Calculadora conversor de unidades	33
Figura 4-28: Mapa de estaciones meteorológicas de <i>Meteonorm</i>	34
Figura 4-29: Modificación de datos <i>Meteonorm</i>	34
Figura 4-30: Datos obtenidos en <i>Meteonorm</i>	35
Figura 4-31: Exportar datos <i>Meteonorm</i>	35
Figura 4-32: Ubicación de la estación meteorológica en la ENM [8].....	36
Figura 4-33: Arduino UNO	36
Figura 4-34: <i>SparkFun Sunny Buddy</i>	37
Figura 4-35: <i>SparkFun Electric Imp Shield</i>	37
Figura 4-36: <i>SparkFun Weather Shield</i>	38
Figura 4-37: RJ11 6-Pin Connector	38
Figura 4-38: Arduino Stackable Header Kit - R3	39
Figura 4-39: <i>Electric Imp</i>	39
Figura 4-40: <i>Solar Cell Large</i>	40
Figura 4-41: <i>Polymer Lithium Ion Battery</i>	40
Figura 4-42: Veleta de viento	40
Figura 4-43: Soporte superior para anemómetro y veleta	41
Figura 4-44: Soporte medio para pluviómetro	41
Figura 4-45: Barra soporte principal	41
Figura 4-46: Distribución de sensores de la estación	41
Figura 4-47: Distribución de sensores de la estación 2	42
Figura 4-48: Modificación <i>SparkFun Electric Imp Shield</i>	42
Figura 4-49: Montaje del Arduino.....	43
Figura 4-50: Soldadura de los pines	43
Figura 4-51: Alimentación de la estación meteorológica.....	43
Figura 4-52: Software Arduino UNO.....	44
Figura 4-53: Selección de placa Arduino UNO en software.....	45
Figura 4-54: Selección de puerto COM3 en software	45
Figura 4-55: Gestionar librerías Arduino	45
Figura 4-56: Librerías Arduino	46
Figura 4-57: Verificar código Arduino.....	46

Figura 4-58: Subir código Arduino	46
Figura 4-59: Comprobación de comunicación con el Arduino	47
Figura 4-60: Esquema de funcionamiento de la tarjeta <i>Electric Imp</i>	47
Figura 4-61: Tarjeta <i>Electric Imp</i>	47
Figura 4-62: Vincular tarjeta <i>Electric Imp</i> con página web[24].....	48
Figura 4-63: Datos estación en <i>Wunderground</i>	48
Figura 4-64: Interfaz WEB <i>Imp IDE</i>	49
Figura 4-65: Corona del taladro	50
Figura 4-66: Agujero superior para la entrada de radiación solar	50
Figura 4-67: Taladro vertical.....	50
Figura 4-68: Componentes de la estación meteorológica.....	51
Figura 4-69: Distribución de los componentes en el interior de la caja	51
Figura 4-70: Caja para el Arduino.....	52
Figura 4-71: Primera ubicación de la estación meteorológica	52
Figura 4-72: Segunda ubicación de la estación meteorológica	53
Figura 4-73: Distancia entre Puente de Rande y patrullero Tabarca [8]	54
Figura 4-74: Datos de la plataforma del puente de Rande [6].....	54
Figura 4-75: Datos de temperatura del agua de mar en tablas y gráficos [6].....	54
Figura 4-76: Rosa de los vientos	56
Figura 4-77: Software <i>Weather</i> de <i>EnergyPlus</i>	57
Figura 4-78: Archivos convertidos con <i>Weather</i>	58
Figura 4-79: Simulación <i>Trnsys</i> para <i>Weather</i>	58
Figura 4-80: Type 56 a editar para leer archivos .epw	58
Figura 4-81: Escudo patrullero Tabarca P28 [32]	59
Figura 4-82: Patrullero Tabarca P-28 [32]	60
Figura 4-83: Patrullero Tabarca P-28 en su puerto base [32].....	61
Figura 4-84: Marca de Plimsoll franco bordo [38].....	63
Figura 4-85: Curvas hidrostáticas Clase Anaga [32].....	64
Figura 4-86. Modelo en <i>SketchUp</i> para una zona térmica de 300 m ³	65
Figura 4-87: Ubicación del patrullero Tabarca [8].....	65
Figura 4-88: Simulación de un volumen en <i>Trnsys</i>	66
Figura 4-89: Datos meteorológicos de la simulación	66
Figura 4-90: <i>TRNBuild</i>	67
Figura 4-91: Selección de material en <i>TRNBuild</i>	67
Figura 4-92: Propiedades Acero de la biblioteca <i>TRNBuild</i>	68

Figura 4-93: Climatización de la zona con temperaturas de consigna	69
Figura 4-94: Aplicación de la climatización	69
Figura 4-95: Resultados simulación de un volumen	70
Figura 4-96: Resultados de intercambio de calor de la simulación de un volumen	71
Figura 4-97: Resultados de temperaturas de la simulación de un volumen	71
Figura 4-98: Modelo en <i>SketchUp</i> para dos zonas térmicas de 300 m ³	73
Figura 4-99: Resultados simulación de dos volúmenes	74
Figura 4-100: Resultados de intercambio de energía de la simulación de dos volúmenes	74
Figura 4-101: Resultados de temperaturas de la simulación de dos volúmenes	75
Figura 4-102: Volumen de <i>SketchUp</i> simulando al patrullero Tabarca [8].....	78
Figura 4-103: Simulación Temperatura del mar en <i>Trnsys</i>	78
Figura 4-104: Configuración type 9b	79
Figura 4-105: Introducir dato de temperatura del mar en la simulación	79
Figura 4-106: Resultados de temperaturas para verano	80
Figura 4-107: Resultados de temperaturas para invierno	80
Figura 4-108: Sensor de temperatura y humedad ds1923-f5# [45].....	83
Figura 4-109: Sensor de temperatura y humedad ds1923-f5# [45].....	83
Figura 4-110: Sensor de temperatura ds1922l-f5# [45]	84
Figura 4-111: Usb (ds9490b#)	84
Figura 4-112: Sensores colocados en el Tabarca	85
Figura 4-113: Distribución de sensores	85
Figura 4-114: Ubicación de sensores a bordo del Tabarca.....	86
Figura 4-115: Software <i>OneWireViewer</i>	87
Figura 4-116: Programación de un sensor.....	87
Figura 4-117: Gel de sílice con indicador de color naranja [46].....	90
Figura 4-118: Bolsa sumergible con gel de sílice	90
Figura 4-119: Cápsula Antiagua DS9107 con sensor	91
Figura 4-120: Cápsula Antiagua DS9107	91
Figura 4-121: Bolsa sumergible con cápsula en su interior	91
Figura 4-122: Datos de temperatura del mar de MeteoGalicia [6].....	93
Figura 4-123: Datos de temperatura del mar de sensores	93
Figura 5-1: Modelo patrullero Tabarca en <i>SketchUp</i>	96
Figura 5-2: Definición de paredes en la obra viva	96
Figura 5-3: Modelo completo patrullero Tabarca en <i>SketchUp</i>	97
Figura 5-4: Simulación del patrullero Tabarca en <i>Trnsys</i>	97
Figura 5-5: Relación entre type 9a y type Tabarca.....	98

Figura 5-6: <i>Inputs</i> type 65c.....	98
Figura 5-7: Conversor de unidades.....	99
Figura 5-8: Configurar parámetros type 9a	99
Figura 5-9: Asignar materiales a los cerramientos	100
Figura 5-10: Propiedades de los cerramientos	101
Figura 5-11: Climatización de los volúmenes	101
Figura 6-1: Posición del Tabarca respecto a los edificios próximos.....	107
Figura 6-2: Posición del Tabarca respecto a los edificios próximos.....	108
Figura 6-3: Primera colocación de la estación meteorológica.....	108
Figura 6-4: Segunda colocación de la estación meteorológica	109
Figura 6-5: Datos obtenidos por la estación meteorológica	109

ÍNDICE DE TABLAS

Tabla 1-1: Secuencia de trabajo	11
Tabla 4-1: Patrulleros Clase Anaga [32, 33]	60
Tabla 4-2: Propiedades térmicas del Acero	68
Tabla 4-3: Resumen de resultados de la simulación para un volumen	72
Tabla 4-4: Resumen de resultados de la simulación para dos volúmenes.....	75
Tabla 4-5: Comparación de temperaturas internas (° C)	76
Tabla 4-6: Comparación de necesidades térmicas (W)	77
Tabla 4-7: Análisis de la influencia de la temperatura del mar	80
Tabla 4-8: Resultados simulación con temperaturas del mar	81
Tabla 4-9: Temperatura media del buque por días	88
Tabla 4-10: Temperatura media diaria	88
Tabla 4-11: Temperaturas en función de la hora.....	89
Tabla 4-12: Temperatura máxima y mínima alcanzada	89
Tabla 4-13: Temperaturas diarias medidas con sensores in situ	92
Tabla 4-14: Comparación de datos de temperatura del agua de mar	93
Tabla 4-15: Porcentaje de diferencia entre datos de MeteoGalicia y sensores	94
Tabla 5-1: Materiales empleados en los cerramientos	100
Tabla 5-2: Propiedades de los materiales	100
Tabla 5-3: Resultados de simulación con datos de Meteonorm y datos de agua de mar obtenidos con los sensores.....	102
Tabla 5-4: Resultados de simulación con datos del equipo de investigación GTE de la Universidad de Vigo y datos de agua de mar obtenidos con los sensores.....	103
Tabla 5-5: Diferencias entre diferentes fuentes de datos meteorológicos	104

ÍNDICE DE ECUACIONES

Ecuación 4-1: Temperatura del punto de rocío	55
Ecuación 4-2: Humedad relativa	57
Ecuación 4-3: Calor sensible	70
Ecuación 4-4: Porcentaje de diferencia en datos	76
Ecuación 4-5: Porcentaje de diferencia en datos	81
Ecuación 4-6: Variación de temperatura en tramos horarios	89

1 OBJETIVOS

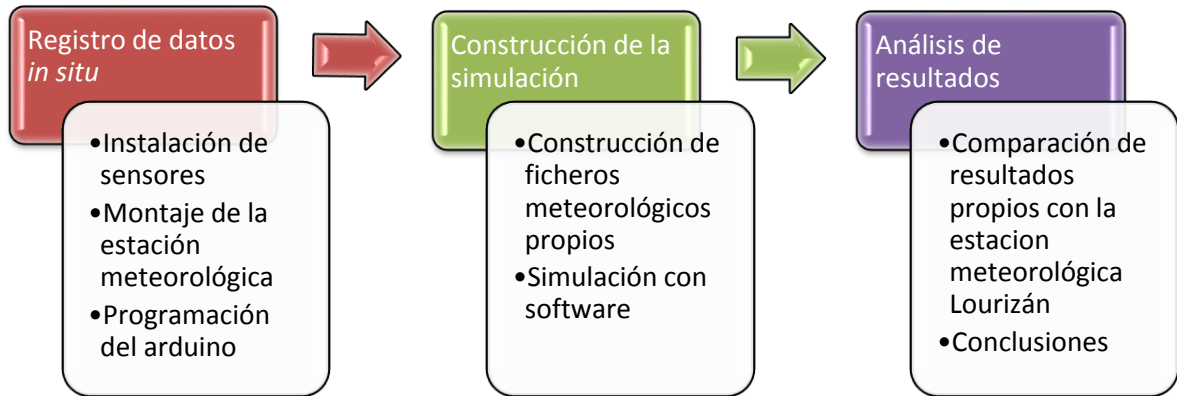
El objetivo principal de este Trabajo Fin de Grado es analizar la influencia de variables meteorológicas en la simulación térmica de buques, contrastando los resultados que se obtienen utilizando bases de datos internacionales (i.e. *Meteonorm*), con mediciones in situ y datos reales de estaciones meteorológicas próximas a la ubicación estudiada, concretamente el muelle Almirante Vierna de la Escuela Naval Militar de Marín.

En el presente trabajo se pretenden conseguir datos meteorológicos más precisos que los obtenidos de *Meteonorm* y/o de la estación más cercana de la red de *MeteoGalicia*. En el primer caso, *Meteonorm*, para estimar las variables meteorológicas, somete a los datos de su red a procesos de interpolación (e.g. Kriging, Thin plate spline, etc.), con lo que siempre se comete un error, que se acaba propagando al resultado final de la simulación. Otra opción muy empleada en la simulación térmica es utilizar los datos de la estación más cercana al punto de análisis, en este caso, una estación de la red de *MeteoGalicia*. Pese a que la red de *MeteoGalicia* es muy amplia, siempre existe una variación entre lo que sucede en el punto de análisis y la medición de la estación de la red, ubicada a varios kilómetros de distancia del punto. Por todo ello, se colocarán sensores de medición de parámetros meteorológicos (e.g. temperatura, velocidad del viento, presión, etc.) en el punto de análisis.

Para conseguir estos objetivos, se empleará software de simulación térmica en edificios (i.e. *Trnsys*), lo que permitirá comparar los datos medidos in situ y los obtenidos de redes de estaciones meteorológicas. Estas simulaciones se harán en base a las características de un buque de la Armada Española, concretamente, el patrullero *Tabarca*, que, al tener su puerto base en la Escuela Naval Militar, servirá de punto para la toma de datos.

La secuencia lógica de trabajo que se seguirá para alcanzar estos objetivos se resumen en la Tabla 1-1. Se comenzará por la realización de mediciones de los parámetros a estudiar a bordo del patrullero *Tabarca*. Se construirá la estación meteorológica propia y se programará el Arduino que nos proporcionará los datos medidos en una base de datos en la web. Después de esto, se obtendrán los ficheros meteorológicos medidos in situ con los que se realizarán simulaciones con el software específico. Para finalizar se compararán los resultados de las simulaciones empleando datos meteorológicos medidos in situ y los obtenidos de la red de *MeteoGalicia*.

Tabla 1-1: Secuencia de trabajo



2 INTRODUCCIÓN

2.1 La simulación térmica de edificios

La simulación térmica de edificios permite obtener el comportamiento del edificio, considerando los parámetros físicos (la fachada, orientación o cerramientos), a través del estudio de las principales variables climatológicas, de habitabilidad y energía [1]. El objetivo, por tanto, es conocer durante un periodo de tiempo dado una determinada característica energética o de regulación del sistema, también estimar los consumos de energía, analizar diferentes estrategias de control, analizar diferentes comportamientos de máquinas o sistemas de climatización, observar la evolución de temperaturas o humedades relativas en las diferentes zonas que conforman el edificio, etc. Está cobrando gran importancia, sobre todo, en el estudio para la eficiencia energética.

Se trata, por tanto, de estimar el comportamiento térmico de las diferentes zonas que conforman el edificio en cada instante de tiempo.

Al final de la simulación se conocerá la energía necesaria para climatizar la zona en forma de potencia que con el sistema internacional (SI) se medirá en Vatios ($J/s=W$).

La simulación térmica mediante aplicaciones informáticas tiene en cuenta una serie de procesos de transmisión de calor de forma transitoria, como son: la conducción a través de la estructura del edificio; la convección en las superficies del edificio y en los espacios interiores; la radiación solar absorbida, reflejada y transmitida por cada elemento del edificio; y por último, los incrementos térmicos internos [2].

En el presente trabajo, se realizará la simulación térmica de un buque, considerado como construcción flotante que debe ser lo más eficiente posible, para reducir gastos de combustible en climatización y, así, obtener mayor autonomía. Se estudiarán para la simulación térmica, los materiales de los cerramientos, así como las infiltraciones, realizando el mismo procedimiento que se emplearía para una simulación de un edificio.

2.2 Importancia de las variables meteorológicas en la simulación térmica

La simulación energética tiene en cuenta no sólo la geometría y materiales del edificio, sino también las condiciones climatológicas exteriores, y el uso del edificio (niveles de ocupación y horarios de funcionamiento, cargas interiores debidas al uso previsto, etc.). En los últimos años los modelos de calibración energética han ganado importancia. Estas calibraciones permiten un ahorro energético al obtener mediciones más precisas en el lugar de construcción, realizando de esta manera cálculos más exactos [3].

En el proceso de simulación energética de edificaciones, el clima cumple un papel fundamental; para saber cuál es su comportamiento se tiene que recurrir a datos meteorológicos en zonas cercanas donde se realiza el proyecto.

El concepto de “zona térmica” es necesario para la simulación energética, representa un volumen de aire a una temperatura uniforme delimitado por superficies en las que se produce una transferencia y almacenamiento de calor [4].

Las principales variables meteorológicas que deben tenerse en cuenta en simulaciones como esta, en las que únicamente estudiaremos la obra viva de un buque, son la radiación solar global, velocidad y dirección del viento, humedad, temperatura ambiente y temperatura del medio en el que se encuentra sumergido, tratándose de agua salada en este caso.

En cuanto a la radiación solar, debemos tener en cuenta que supone la suma de radiación directa, indirecta y reflejada (en la superficie del mar para este proyecto). Deben tenerse en cuenta en caso de un buque, la orientación y forma del mismo y la posición del sol durante el día. La radiación solar se mide en vatios por metro cuadrado (W/m^2).

La radiación solar calienta la tierra, y ésta calienta el aire que está en contacto con su superficie. Cuando el aire se calienta, también se dilata, como resultado su densidad disminuye, por lo cual asciende hasta que su temperatura se iguala con la del aire circundante. De esta manera se genera los vientos, que pueden ir desde lentas brizas hasta grandes huracanes. El aparato que mide la velocidad del viento es el anemómetro, el tipo más común en meteorología es el anemómetro de cazoletas (3 o 4 aspas) en el que se mide la velocidad de rotación, aunque empieza a ser substituido por el anemómetro sónico (se mide la diferencia de velocidad del sonido entre dos emisores/sensores (montando tres pares se mide también la dirección del viento)).

La humedad también es un factor de interés para las simulaciones. La humedad relativa, HR [%], es la proporción de vapor de agua real en el aire comparada con la cantidad de vapor de agua necesaria para la saturación a la temperatura correspondiente. Indica que tan cerca está el aire de la saturación. Se mide en porcentaje entre 0 y 100, donde el 0% significa aire seco y 100% aire saturado.

Por último, las temperaturas tanto del agua como del ambiente y de punto de rocío se miden con termómetros propios o a partir de los datos proporcionados por las estaciones meteorológicas cercanas.

2.3 Obtención de datos meteorológicos

Para el desarrollo de este apartado se ha empleado como referencia la Organización Meteorológica Mundial [5].

La obtención de datos meteorológicos es muy variada dependiendo de la precisión y la escala del terreno que se desee obtener. Para observar los sistemas meteorológicos a escala mundial se emplean satélites meteorológicos, contando en la actualidad con cerca de 160 en órbita, los cuales realizan unas 80 millones de observaciones por día.

Hay dos tipos de satélites meteorológicos:

Los satélites de órbita polar, se encuentran a baja altitud en la órbita entre el Polo Norte y Polo Sur. Con ellos se obtiene información y datos para predicciones meteorológicas.

Los satélites geoestacionarios (Figura 2-1), se encuentran en órbita a altitudes elevadas a la misma velocidad que la rotación de la Tierra. Con ellos se mantiene vigilancia en una zona concreta de la Tierra, proporcionando información sobre formación de nubes, tormentas o incluso incendios.



Figura 2-1: Satélite meteorológico[5]

Los radares Doppler o radares meteorológicos (Figura 2-2) se emplean para prever las precipitaciones –lluvia, nieve, granizo–, etc. Calculan el desplazamiento y la intensidad de la futura precipitación, y observan la evolución de la misma hasta convertirse en tormenta. Un radar Doppler emite ondas radioeléctricas desde su antena, reflejándose estas en las materias de la atmosfera como las gotas de lluvia.

El radar convierte las ondas reflejadas en imágenes para seguimiento y ubicación de las precipitaciones.

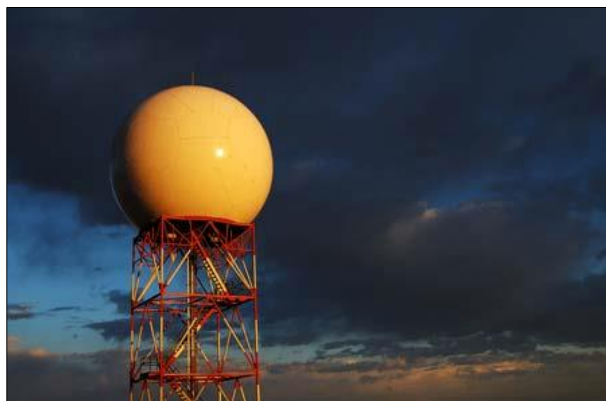


Figura 2-2: Radar Doppler [5]

Una estación meteorológica es una instalación destinada a medir y registrar regularmente diversas variables meteorológicas, cuentan con sensores de alta exactitud y resolución para medir cada variable, entre las marcas más conocidas de estaciones meteorológicas se puede mencionar a *Campbell Scientific*, *Davis Instruments* y *Onset*.

Generalmente la estación meteorológica tiene los siguientes elementos:

- Sensores: que miden velocidad y dirección del viento, temperaturas, presión atmosférica, humedad relativa externa, radiación solar, sensor UV y Pluviómetro.
- Modulo interfaz de sensores (SIM) contiene la electrónica que mide y almacena los datos meteorológicos para transmitirlos a la consola vía cable u ondas de radio (inalámbrico).
- Consola o *datalogger*, encargada de almacenar todas las mediciones que realicen los sensores.
- Software, es necesario para programar la estación meteorológica y descargar los datos almacenados en el *datalogger* para realizar su tratamiento.

Algunos de los aparatos citados se colocan en una garita al abrigo meteorológico, normalmente tipo Stevenson (Figura 2-3). Está hecha de madera con láminas pintadas de color blanco para facilitar la ventilación y, a la vez, proteger a los aparatos de medida de la radiación solar y de la lluvia.



Figura 2-3: Estación meteorológica. Garita Stevenson [5]

Por último, los globos meteorológicos (Figura 2-4) transportan radiosondas en la atmósfera superior. Estas miden las condiciones atmosféricas, como la presión barométrica, la humedad relativa, la temperatura y la dirección y velocidad de los vientos, y envían la información a una estación en tierra firme a través de una radio. La información recopilada permite realizar predicciones meteorológicas a largo plazo. Las radiosondas pueden también expulsarse de una aeronave, las cuales suelen denominarse radiosondas con paracaídas.



Figura 2-4: Globo meteorológico [5]

Se realiza una recopilación diaria de la información meteorológica a partir de satélites meteorológicos, radares Doppler, estaciones meteorológicas, globos meteorológicos y otras fuentes, como aeronaves y buques. Luego se procesa y se transforma en algo comprensible para todos.

Los ordenadores procesan la información basándose en modelos numéricos elaborados por científicos para obtener las diferentes previsiones meteorológicas.

2.3.1 Red de estaciones de MeteoGalicia

Para obtener los datos en una ubicación determinada, a partir de los medidos en estaciones meteorológicas cercanas se realizan interpolaciones. Para estas interpolaciones se emplea una regresión lineal múltiple donde se considera la altitud de la estación, longitud, latitud y orientación del terreno. Los residuos de esta regresión se interpolan con kriging y se suman a los valores de la regresión [6, 7].

La Red de estaciones de MeteoGalicia (Figura 2-5) es un servicio meteorológico regional dependiente de la Consejería de Medio Ambiente, Territorio e Infraestructuras de la Junta de Galicia. Para realizar la predicción diaria ejecuta diferentes modelos numéricos meteorológicos y oceanográficos. Además dispone de una extensa red de observación meteorológica y oceanográfica [6].



Figura 2-5: Red de MeteoGalicia [6]

La red de MeteoGalicia está integrada por un total de 150 estaciones, de las cuales 56 llevan sensores agrometeorológicos y 94 meteorológicos. Todas las estaciones registran temperatura y humedad relativa, pero solo 105 radiación solar global. 98 estaciones incorporan sensor de presión y 67 miden viento meteorológico [7].



Figura 2-6: Distancia entre patrullero Tabarca y estación de Lourizán [8]

La estación más cercana al lugar de estudio, y por tanto los datos que se usarán, es la estación de Lourizán (Figura 2-7), ubicada en $42^{\circ}24'36.0''\text{N}$ de latitud y $008^{\circ}39'36.0''\text{W}$ de longitud a una distancia de 4.07 kilómetros en línea recta del patrullero Tabarca (Figura 2-6).



Figura 2-7: Estación meteorológica Lourizán [9]

Esta estación se encuentra situada a 57 metros de altura y dispone de los siguientes sensores [6]:

- Anemómetro modelo 107H4M del fabricante *Ornytion*: Instrumento para medir la velocidad de circulación de un fluido gaseoso, en especial del viento, en este caso con una precisión de 0.025 m/s. (Figura 2-8).
- Piranómetro modelo 8101 del fabricante *Schenk*: Instrumento meteorológico utilizado para medir de manera muy precisa la radiación solar incidente sobre la superficie de la tierra. (Figura 2-9).
- Pluviómetro modelo 5.4032.35.007 del fabricante *Thies*: Instrumento para medir la cantidad de lluvia que cae en un lugar y en un espacio de tiempo determinados; el agua recogida por él se mide en litros o milímetros por metro cuadrado (Figura 2-10).
- Sonda Horas de Sol modelo CSD 1 del fabricante *Kipp&Zonen*: Instrumento meteorológico para medir la radiación directa del sol (Figura 2-11).
- Barómetro modelo PTB110 del fabricante *Vaisala*: Instrumento para medir la presión atmosférica (Figura 2-12).
- Sonda de temperatura y humedad modelo HMP155 del fabricante *Vaisala*: mide dichos parámetros del aire (Figura 2-13).



Figura 2-8: Anemómetro [6]



Figura 2-9: Piranómetro [6]



Figura 2-10: Pluviómetro [6]



Figura 2-11: Sonda horas de sol [6]



Figura 2-12: Barómetro [6]



Figura 2-13: Sonda de temperatura y humedad [6]

3 ESTADO DEL ARTE

3.1 Trabajos existentes sobre simulación térmica

Durante los últimos años han adquirido gran importancia las simulaciones térmicas de edificios para obtener la máxima eficiencia térmica en nuevas construcciones o rehabilitación de ellas. Se busca con estas simulaciones térmicas el ahorro de energía y el análisis de los cerramientos en diferentes tipos de construcciones. La Calificación Energética de Edificios tiene su origen en la Directiva 2002/91/CE de eficiencia energética de edificios. Supone un aumento de los reglamentos sobre calidad térmica de los edificios. Propone el reconocimiento oficial como edificios de alta eficiencia energética a los edificios cuyo consumo energético sea inferior al que se indica en la normativa [10].

El estudio de la envolvente térmica de edificios mediante programas de simulación se ha transformado en una herramienta indispensable para enfrentar problemas en el diseño de edificios energéticamente eficientes. El avance realizado en este aspecto se refleja en la gran variedad de programas de simulación disponibles actualmente en el mercado [11].

Existen diversos artículos en los que se reflexiona sobre los factores influyentes en el ahorro energético y búsqueda de la eficiencia térmica en edificios, constituyendo estos un papel importante para la conservación del medio ambiente y reducción del calentamiento global [12].

3.2 Trabajos existentes sobre simulación térmica en buques

La simulación energética avanzada de edificios permite calcular el comportamiento del edificio a través de la evolución de las principales variables de clima, energía y habitabilidad, considerando todos los parámetros físicos que lo componen (forma y orientación, fachada, cerramiento, etc.).

La simulación permite también tener en cuenta aspectos como la ocupación y el uso de cada espacio o los datos ambientales y climáticos específicos del lugar, y evaluar y optimizar múltiples aspectos constructivos como [13]:

- los niveles de bienestar térmico (temperatura radiante, operativa...) y bienestar lumínico (factor de luz diurna, riesgo de deslumbramiento...)
- el comportamiento de las características constructivas de la envolvente térmica del edificio (geometría, orientaciones, inercia térmica, aislamiento, acristalamiento, protecciones solares...)
- las instalaciones de climatización, iluminación, ACS, energías renovables, etc.
- el cumplimiento normativo y la certificación de eficiencia energética de los edificios;

- el aprovechamiento potencial de la ventilación natural, la disposición de aperturas, el efecto chimenea, la ventilación cruzada, la iluminación artificial, la ventilación, la climatización, etc.

En cambio, las simulaciones térmicas en buques no han sido muy estudiadas, por lo que se trata de un ámbito relativamente novedoso.

Se conocen estudios de eficiencia energética en el sector naval como el estudio para analizar el comportamiento de un sistema de micro-generación de un motor Stirling que porta un barco de vela, analizando las condiciones meteorológicas y la influencia de la temperatura de la superficie del agua de mar en diferentes ubicaciones [14]. Otros estudios del sector naval se enfocan a ahorro de energía en barcos pesqueros debido al aumento del precio del petróleo, y búsqueda de la rentabilidad de las flotas pesqueras o transporte marítimo como por ejemplo “Buques de Pesca y Eficiencia Energética: Proyecto Peixe Verde” y “Medidas para mejorar la eficiencia energética en el transporte marítimo” [15, 16].

Otro estudio con el software de simulaciones térmicas de edificaciones aplicado al sector de la Armada Española es el “Estudio de las necesidades energéticas de un buque en diferentes ubicaciones” [17]. Por la escasez de estudios de eficiencia energética en los buques de la Armada Española se decide comenzar este estudio, sobre la influencia de las variables meteorológicas en las simulaciones térmicas, empleando como caso de estudio el patrullero Tabarca.

3.3 Software utilizado en la simulación térmica

El coste de ordenadores personales con potencias de cálculo y capacidad de almacenamiento era, accesible a pocas instituciones hace pocos años. Durante los últimos años ha avanzado esta tecnología, permitiendo a gran parte de la población a disponer de un ordenador personal donde trabajar, obtener información o incluso utilizar software complejo como es el de simulación.

En Estados Unidos este tipo de software surgió como resultado de la inversión lenta pero constante de la administración americana desde 1970. Muchos de ellos se empezaron a programar con lenguaje como el *FORTRAN*, que han evolucionado hacia un lenguaje más similar a C++ o JAVA y están orientados a objetos. Otros programas como el *TRNSYS* (*Transient Simulation of Systems*) de la Universidad de Wisconsin-Madison (EE.UU.) tuvieron y tienen un ámbito de existencia universitario y no estaban orientados inicialmente hacia la productividad o la industria. Durante los últimos años se ha comenzado a aplicar en todos los ámbitos, educación, investigación y construcción. Otro software como Energy-Plus surge de la unión de software como DOE2 y BLAST [18].

Europa no dispone de una organización que centralice el esfuerzo para el desarrollo de este tipo de software. Lo que existe son diversos programas para diferentes empleos: comerciales, libres, código abierto, etc. Existen empresas que producen programas comerciales para la introducción del modelo-D de forma cómoda en programas americanos de código abierto. Por ejemplo el IISIBAT, del instituto para la edificación francés CSTB que utiliza como núcleo de cálculo el TRNSYS, o el “*Design Builder*” en Inglaterra, que ayuda a generar el fichero .idf que usa el *EnergyPlus* americano.

Empleando el software de este Trabajo de Fin de Grado existen gran variedad de estudios y trabajos, siendo la mayor parte de ellos, del estudio y mejora de la eficiencia energética de edificaciones en diversas ubicaciones como “Procedimientos y aspectos de la simulación de instalaciones térmicas en edificios” [18].

4 METODOLOGÍA

4.1 Software empleado

4.1.1 SketchUp

SketchUp (o *Trimble SketchUp*) es un programa de diseño gráfico y modelado en (3D) tres dimensiones. Se emplea en entornos de arquitectura, ingeniería civil, diseño industrial, diseño escénico, videojuegos o películas. Fue desarrollado por *@Last Software*. Esta empresa fue adquirida por Google en 2006, y vendida en 2012 a *Trimble* [19].

En el presente trabajo se ha empleado *SketchUp 8* para conseguir la compatibilidad con el *Trnsys17*.

Se instala un plug-in que crea una barra de herramientas del programa de simulación térmica *Trnsys* (Figura 4-1). Esto permite la creación de zonas térmicas, generando una compatibilidad entre los diferentes software para facilitar la tarea al usuario [17].

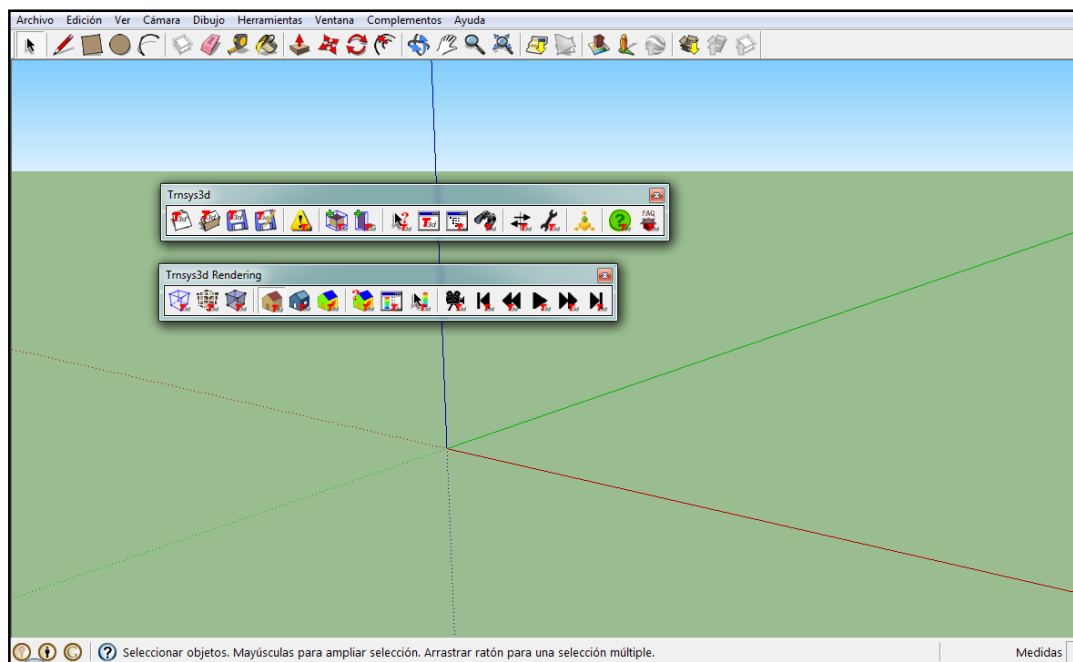


Figura 4-1: SketchUp con plug-in Trnsys

Los botones principales para la creación de una simulación térmica son los siguientes:

Crear nuevos archivos y guardarlos (Figura 4-2).



Figura 4-2: Barra de Herramientas. Crear y guardar archivo

Crear una nueva zona térmica con el boton de la Figura 4-3, se creará una zona cuando cambien las condiciones de envolvente.



Figura 4-3: Barra de Herramientas. Crear zona térmica

Crear una superficie y extruirla para obtener el volumen deseado con las opciones de la Figura 4-4 y Figura 4-5.



Figura 4-4: Barra de Herramientas. Crear superficie



Figura 4-5: Barra de Herramientas. Extruir superficie

Los siguientes botones se emplean para obtener y modificar información sobre las diferentes zonas térmicas y paredes (Figura 4-6, Figura 4-7 y Figura 4-8).



Figura 4-6: Barra de Herramientas. Información

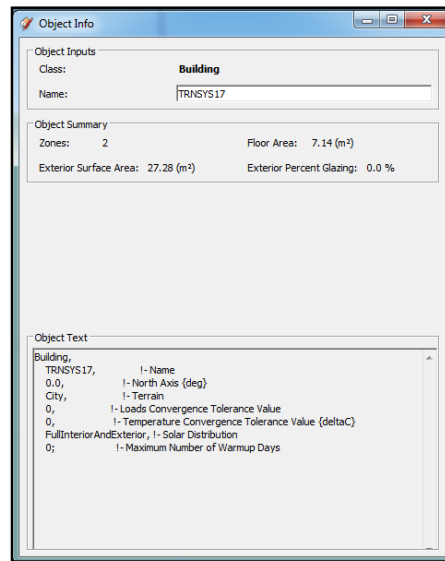


Figura 4-7: Información del objeto

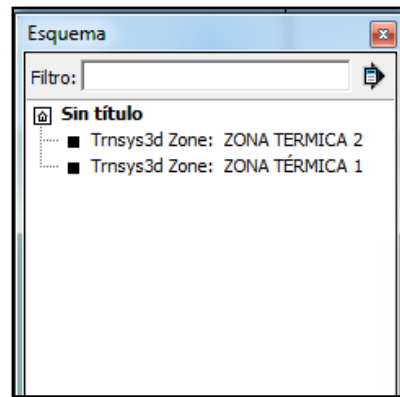


Figura 4-8: Esquema de zonas térmicas

Existen diferentes tipos de cerramientos. En lo que nos debemos fijar primero es en el prefijo tenemos tres:

- EXT Permanecerá en contacto con el aire. Es un cerramiento exterior.
- ADJ Será un cerramiento que está en contacto con otra zona térmica. Es un cerramiento adyacente.
- BND Tendrá condiciones de contorno propias (se seleccionará este tipo cuando la superficie esté en contacto con agua).

Los sufijos indicarán si es pared, ventana, techo, etc. Se asignará *WALL* a las paredes de la edificación, mientras que al techo se le asignará *ROOF* y al suelo *FLOOR*.

Los pasos una vez conocidos los elementos principales del software serán:

Crear una zona térmica y pulsar el botón Intro hasta conseguir que las líneas que indican la zona térmica aparezcan en línea discontinua como se muestra en la Figura 4-9.

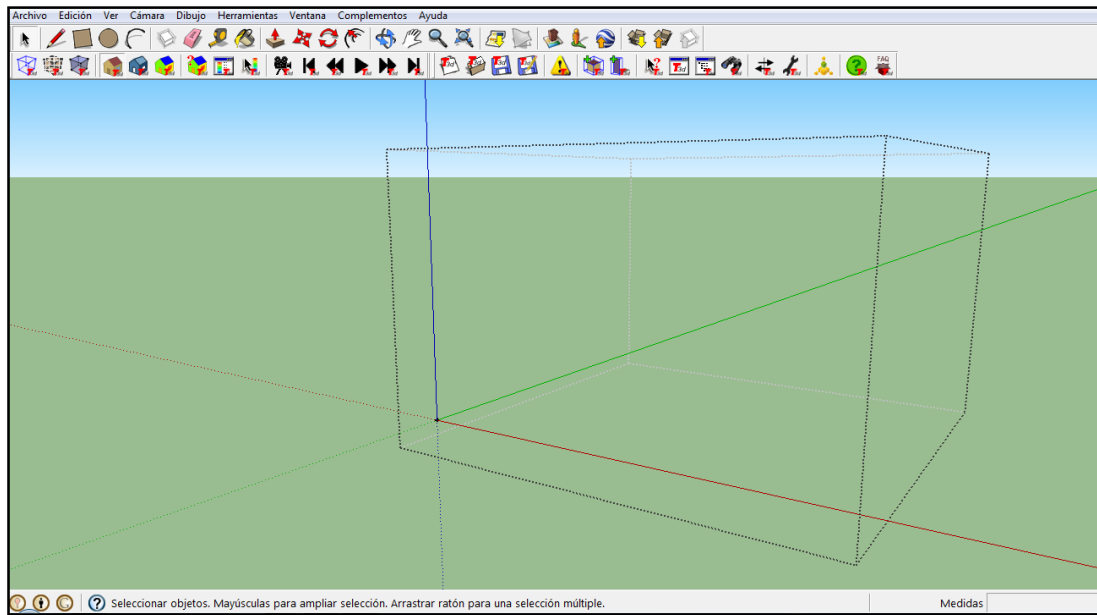


Figura 4-9: Crear una zona térmica en SketchUp

Se debe crear una superficie con las dimensiones deseadas, sin importar si se encuentra dentro o fuera del cubo que representa la zona térmica, ya que aunque visualmente no lo parezca, siempre pertenecerá a esa zona térmica (Figura 4-10).

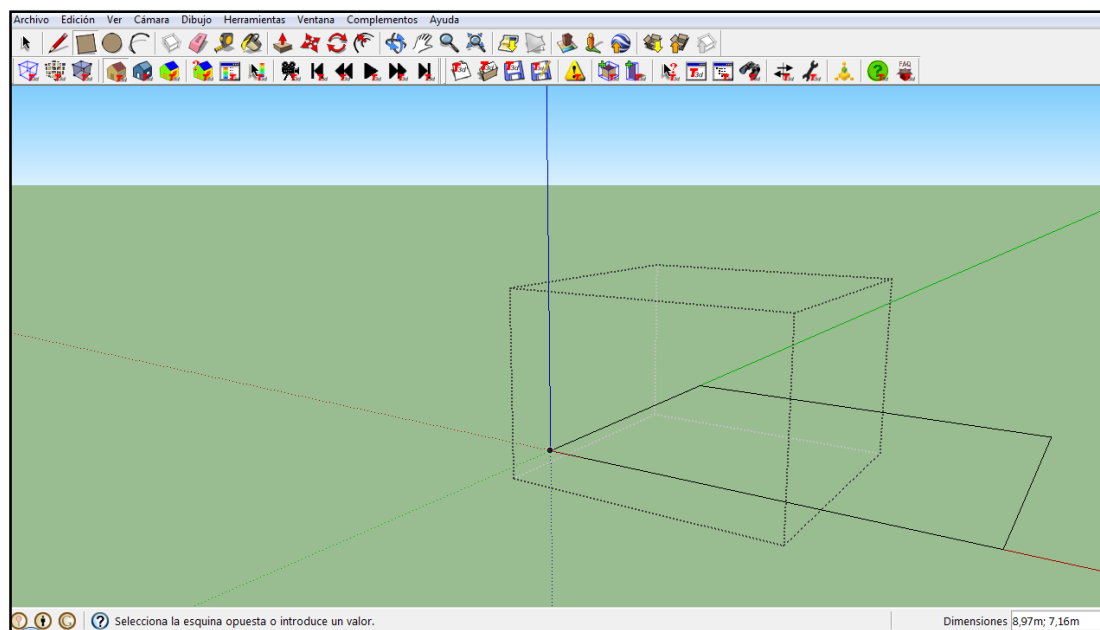


Figura 4-10: Crear superficie en SketchUp

Se debe extruir la superficie como se puede observar en la Figura 4-11 hasta conseguir la edificación deseada. Para que sea más claro, se debe cambiar el nombre del bloque creado seleccionando todo el bloque y pulsando el botón de información de la barra de herramientas.

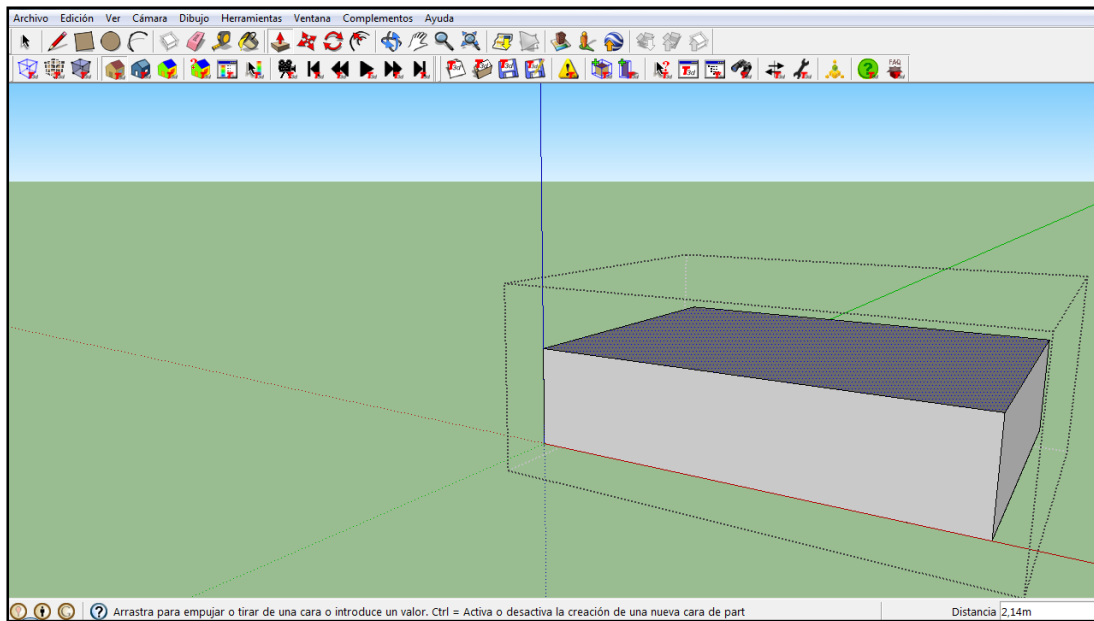


Figura 4-11: Extruir superficie para crear un bloque

Se asignará tipo de pared haciendo doble clic en las paredes hasta que aparezcan con una malla de puntos (Ampliación mostrada en la Figura 4-12).

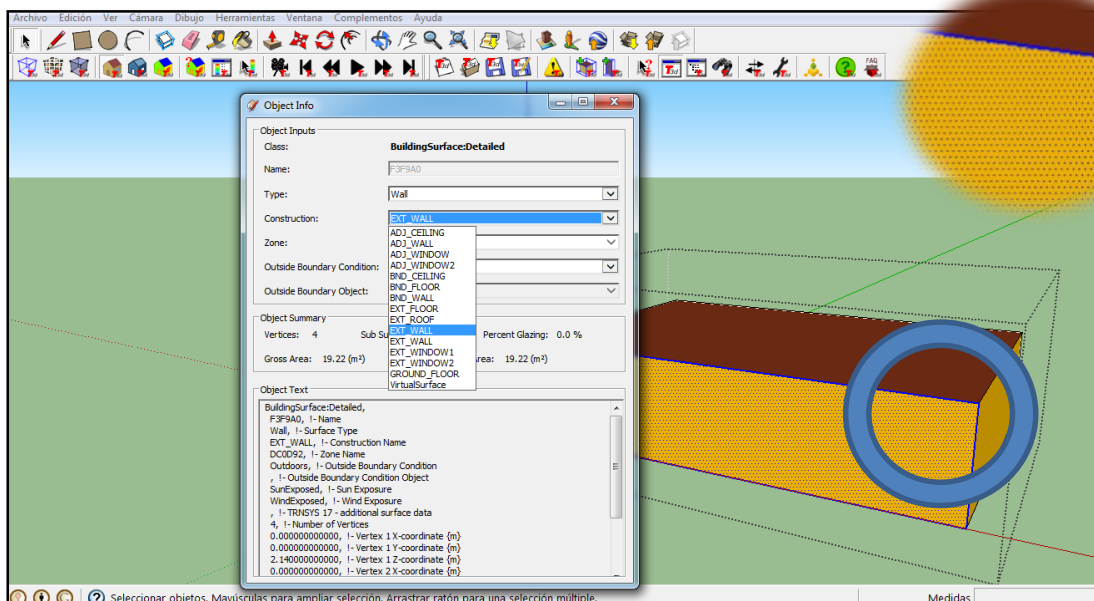


Figura 4-12: Asignar tipo de pared en SketchUp

Una vez creado el modelo y asignado el tipo de pared se debe exportar a *Simulation Studio* de *Tnsys* para continuar con las simulaciones.

4.1.2 Trnsys

4.1.2.1 Descripción del programa

Trnsys es un programa de simulación utilizado principalmente en el campo de la simulación térmica de edificios, simulación de sistemas solares pasivos y diseño solar activo. Se trata de un software desarrollado por la Universidad de Wisconsin. Uno de sus usos originales fue el de realizar la

simulación dinámica del comportamiento de un sistema de agua caliente solar para un año típico meteorológicos a fin de que se pueda comprobar los ahorros de costes a largo plazo de estos sistemas [20, 21].

Trnsys se compone de dos partes, la primera es el motor que lee y procesa los archivos de entrada. Éste proporciona utilidades que (entre otras cosas) determinan las propiedades termo físicas, matrices inversas, realizan regresiones lineales, e interpolar archivos de datos externos. El segundo componente de *Trnsys* es una extensa biblioteca de componentes que incluye aproximadamente 150 modelos que van desde las bombas a edificios multizona, turbinas de viento para electrolizadores, procesadores de datos meteorológicos a las rutinas de la economía, y el equipo básico HVAC a tecnologías de vanguardia emergentes. Los modelos se construyen de tal manera que los usuarios pueden modificar los componentes existentes o escribir su propio, que se extiende las capacidades del medio ambiente.

En el presente trabajo utilizamos la versión *Trnsys 17*.

Simulation Studio es un paquete de simulación completa que contiene varias herramientas, desde programas de motor de simulación y programas de conexión gráfica con software de trazado y hoja de cálculo. Es una herramienta integrada que puede utilizarse desde el diseño de un proyecto para su simulación [20].

TRNBuild es una interfaz para crear y editar toda la información de geometría no requerida por el modelo de edificio de *Trnsys*. *TRNBuild* permite la flexibilidad extensa del usuario en la edición de propiedades del material pared y la capa, crear perfiles de ventilación e infiltración, sumando beneficios, definición de suelos y techos radiantes, posicionamiento ocupantes para comodidad y cálculos [20].

4.1.2.2 Entorno de trabajo

Para comenzar a trabajar con *Trnsys* debemos abrir en *Simulation Studio* un archivo *3D Building Project* (Figura 4-13), donde seleccionaremos el archivo creado en *SketchUp*.

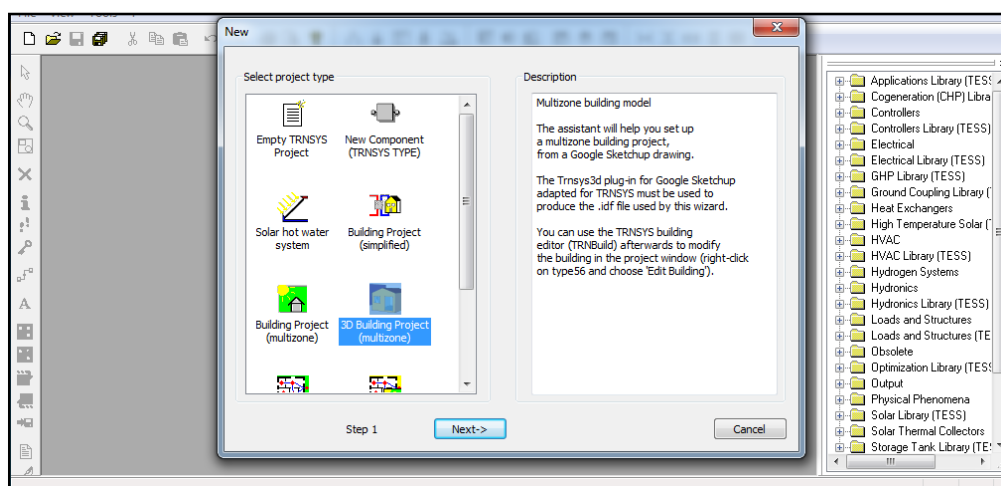


Figura 4-13: *Simulation Studio de Trnsys*

Se abrirá una pantalla como la Figura 4-14 con diferentes types, y debemos editar principalmente el type *Building* para realizar la simulación. Se podrán añadir diferentes types en función de la tarea que quieras desarrollar, seleccionando en las carpetas de la derecha de la pantalla el que se desee.

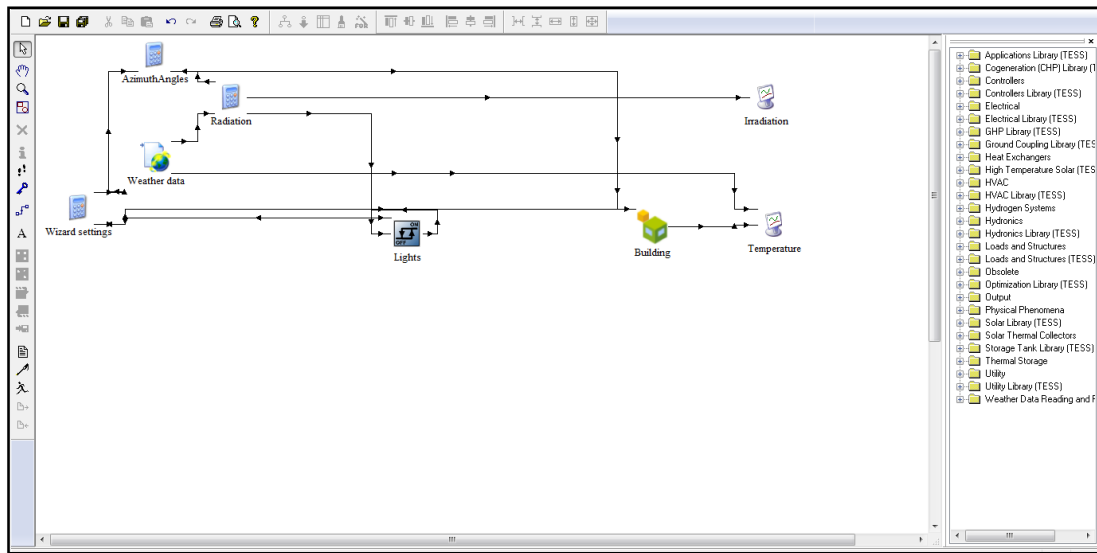


Figura 4-14: Types Simulation Studio

Se debe seleccionar el periodo de horas que se desean simular, para ello empleamos el botón marcado con un círculo rojo en la Figura 4-15.

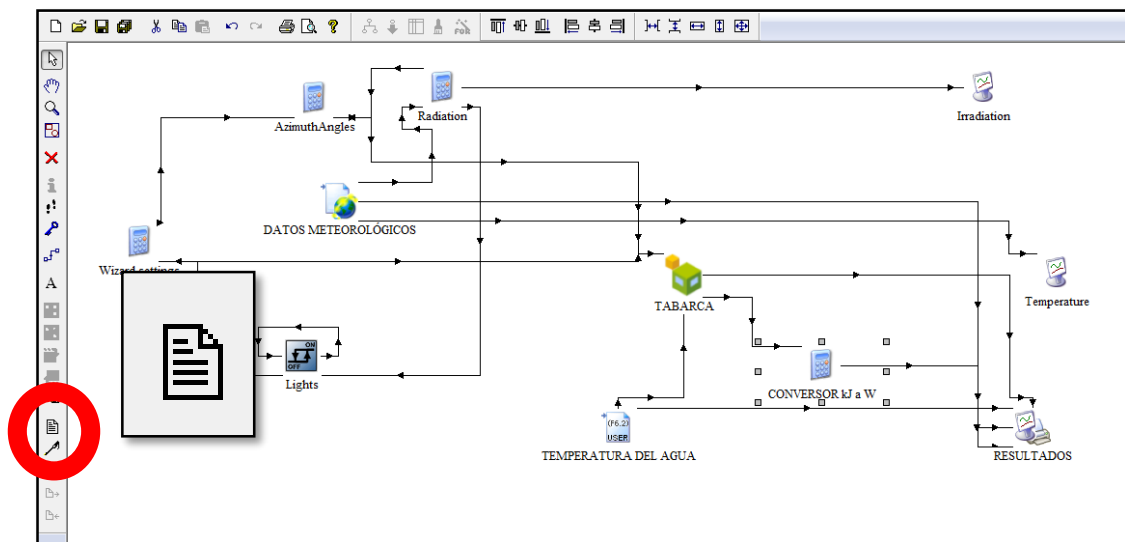


Figura 4-15: Selección del periodo de simulación

Una vez se tenga la simulación con los types deseados se debe editar el *Building*. Para ello, se emplea el *TRNBuild*.

En la documentación de *TRNBuild* se dice que:

“Este componente modela el comportamiento térmico de un edificio dividido en diferentes zonas térmicas. Para poder emplear este componente se debe ejecutar un programa externo de preprocesamiento. *TRNBuild* lee y procesa un archivo que contiene la descripción del edificio y genera dos archivos que usará el type 56 durante la simulación de *Trnsys*. El archivo que contiene la descripción del edificio procesado por *TRNBuild* puede ser generado por el usuario con cualquier editor de texto o con el programa interactivo *TRNBuild*” [21].

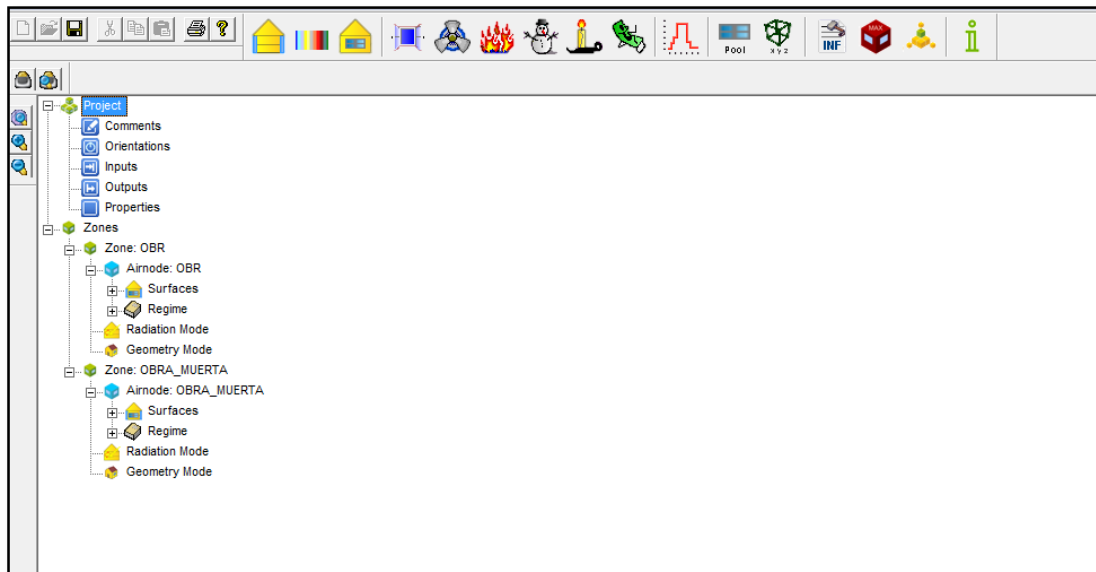


Figura 4-16: TRNBuild

Se debe seleccionar la zona que se desea editar, seleccionando una de las zonas como las mostradas en la Figura 4-16 y con los diferentes botones editar lo que se desee, pudiendo seleccionar tipo y grosor de los materiales de las paredes, sistemas de calefacción y refrigeración o incluso ocupación de personas de la zona que se está editando (Figura 4-17).

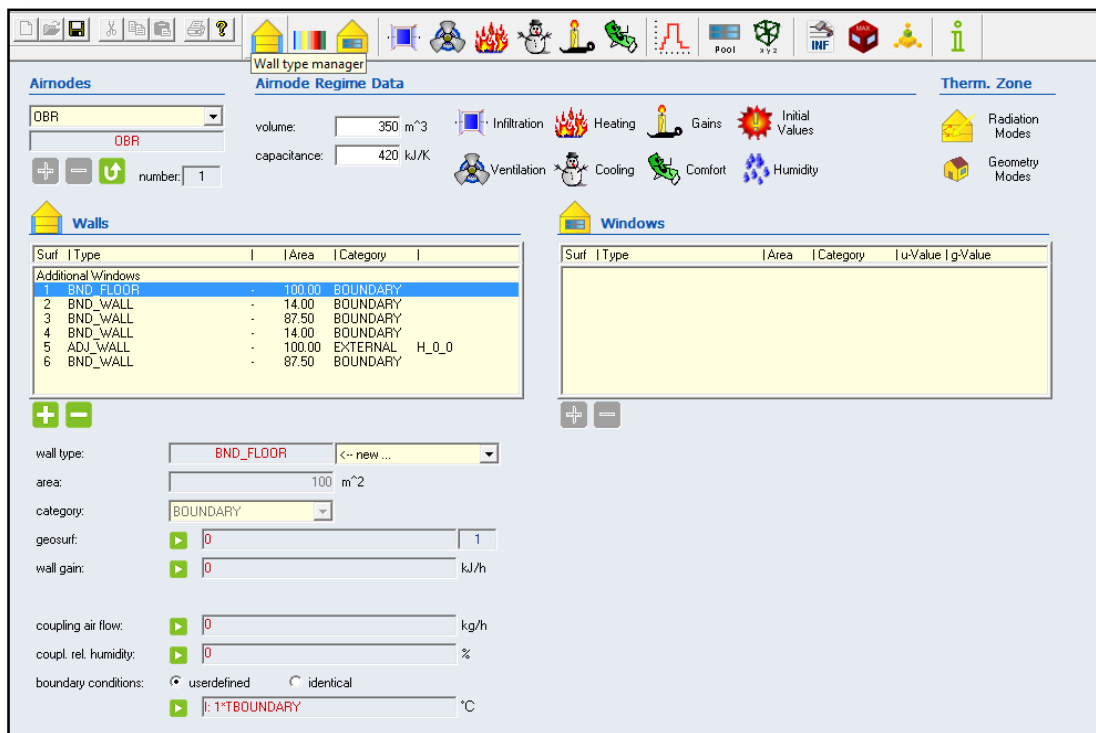


Figura 4-17: Configuración de la zona en TRNBuild

4.1.2.3 Types utilizados

Cada uno de los iconos (Type) puede ser interpretado como un componente que modela parte de la simulación. Se puede editar el nombre de todos los types traduciendo en caso de que se desee los nombres. Estos componentes contienen la siguiente información [20]:

- **Parámetros:** Configuran el comportamiento del type. Ejemplos: potencia de una caldera, rendimiento del radiador, número de superficies, línea a partir de la que lee el archivo de texto.
- **Entradas:** Los inputs son parámetros que pueden variar a lo largo de la simulación. El usuario puede especificar un valor o conectarlas con las salidas de otro componente.
- **Salidas:** Los outputs son valores generados por los componentes. Se pueden enlazar a las entradas de otros componentes, se pueden imprimir, representar en pantalla, etc.
- **Otros Parámetros:** Como pueden ser los nombres de archivos empleados.

Los types que se emplearán en las simulaciones son los que se explicarán a continuación. Además de estos types seleccionados por el usuario, la simulación incluye por defecto, algunos ya configurados que no es necesario modificar, entre ellos *Radiation*, *Wizard Setting*, *Lights* y *Azimet Angels* que funcionan vinculados al type 15-2 (Figura 4-18). Otros types proporcionan gráficos de resultados como *Irradiation* o *Temperatures*.

4.1.2.3.1 Type 15-2: Weather Data



Figura 4-18: Type 15-2: Datos meteorológicos

De la documentación de *Trnbuild*, se puede definir al type 15-2 (Figura 4-18) como un componente que sirve para la lectura de datos a intervalos de tiempo regulares a partir de un archivo de datos externo, obtenido en este caso de *Meteonorm*, siendo capaz de ponerla a disposición de otros componentes de *Trnsys*. Este Este type lee archivos TMY (*Typical Meteorological Year*) y TM2 (*Typical Meteorological Year Version 2*), entre otros.

Se debe introducir como *External File* el archivo meteorológico que se desea emplear en el menú del type mostrado en la Figura 4-19.

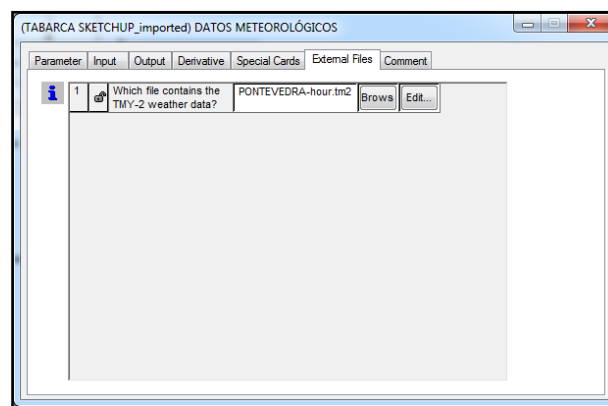


Figura 4-19: Archivo externo con datos meteorológicos.

4.1.2.3.2 Type 9b: Temperatura del agua de mar

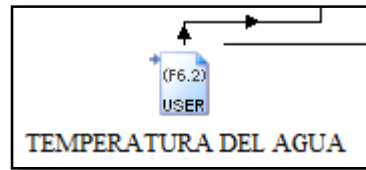


Figura 4-20: Type 9b: Temperatura del agua de mar

Se emplea este type (Figura 4-20) para introducir como Input los datos de temperatura del mar. Se debe crear un archivo de texto con las temperaturas del agua con la frecuencia que se desee (introducirlo como *External File*), configurándola después en los parámetros del type (Figura 4-21). Es importante en el momento de crear el archivo .txt, añadir al final de los datos que se desean introducir en la simulación el primer dato de temperatura del archivo, para en caso necesario que el type repita la lectura del archivo.

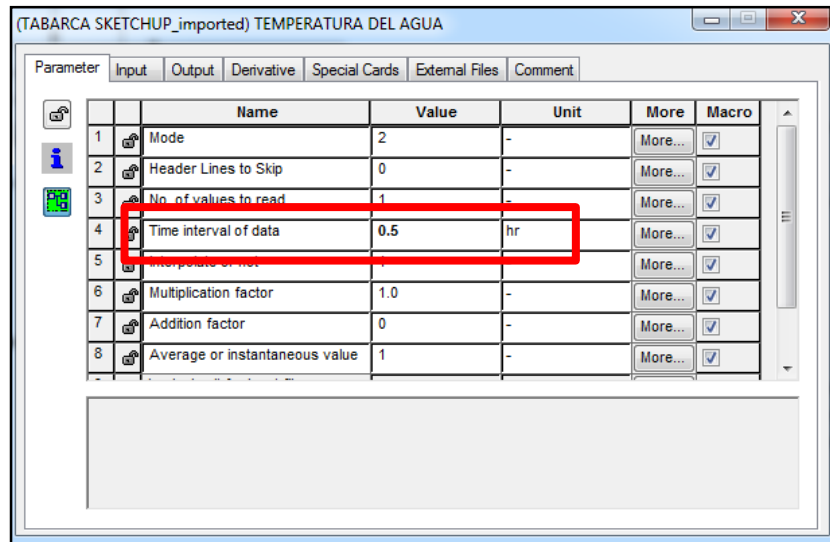


Figura 4-21: Frecuencia de lectura de datos del type 9b

Este type se debe unir con el type principal de la simulación (*Building*), realizando la unión entre *Output 1* y *TBOUNDARY*, que anteriormente se ha configurado en *SketchUp* en las paredes que estarán en contacto con el agua.

4.1.2.3.3 Type 56: Building

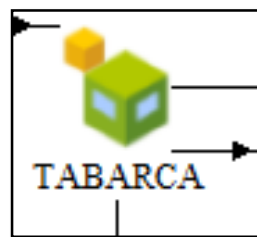


Figura 4-22: Type 56: Building

Este type (Figura 4-22) es el principal de la simulación, ya que introduce el archivo creado en *SketchUp* como *External File* (Figura 4-23) y permite a través del *TRNBuild* modificar los materiales y

grosos de las paredes, así como calefacción y refrigeración de la edificación entre otras cosas. Debe establecerse la unión como *Inputs* con los diferentes types que por defecto existen en la simulación, además del type 9b de temperatura del agua y los *Outputs* con los types de impresión de resultados (type 65c).

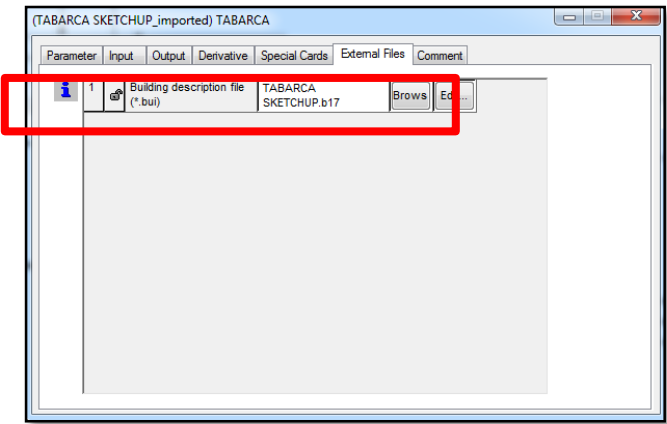


Figura 4-23: *External File* del type 56

4.1.2.3.4 Type 65c: *Online Plotter*



Figura 4-24: Type 65c: *Online Plotter*

Este type (Figura 4-24) se emplea para obtener los resultados de la simulación con los datos que se configuren como *Inputs* en el menú del type (Figura 4-25). Proporciona los resultados a modo de gráfica y como archivo .txt. Permite variar los ejes, nombres y de más características para las gráficas de resultados. Para obtener los resultados en un archivo .txt, debe crearse previamente e introducirlo como *External File*.

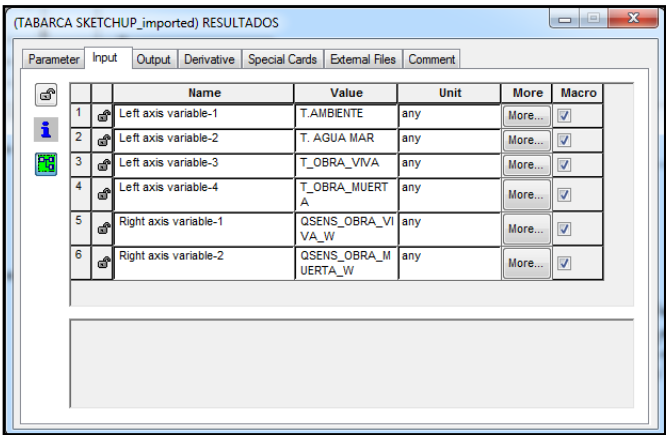


Figura 4-25: *Inputs* del type 65c

4.1.2.3.5 Type Conversor de unidades

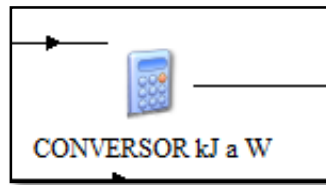


Figura 4-26: Conversor de unidades

Estos types se emplean para realizar operaciones matemáticas con los datos de la simulación para obtener los resultados en las unidades deseadas (Figura 4-26). En este caso, se realiza una conversión de unidades mostrada en la Figura 4-27 pasando de kilojulios (kJ) a vatios (W) y realizando la unión del type 56 con los resultados en vatios del type 65c.

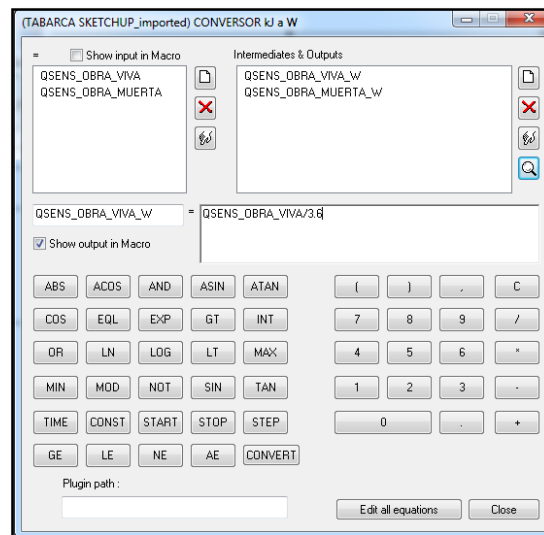


Figura 4-27: Calculadora conversor de unidades

4.1.3 Meteonorm

Tal y como indica el manual del programa, *Meteonorm* es una base de datos meteorológica que contiene datos climatológicos para aplicaciones solares de ingeniería en diferentes lugares del mundo como se puede apreciar en la Figura 4-28. Los resultados son años típicos generados estocásticamente por medio de interpolación a largo plazo. Es un software principalmente empleado para realizar cálculos sobre radiación solar en las diferentes ubicaciones [22, 23].

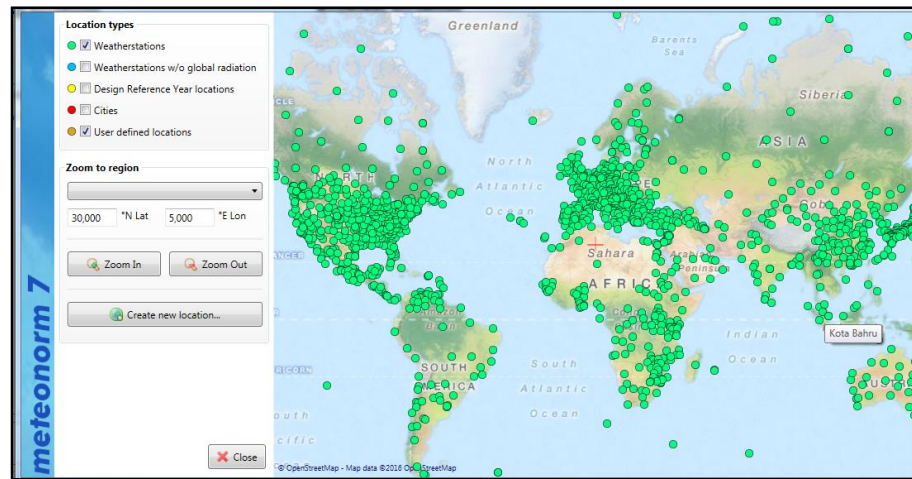


Figura 4-28: Mapa de estaciones meteorológicas de Meteonorm

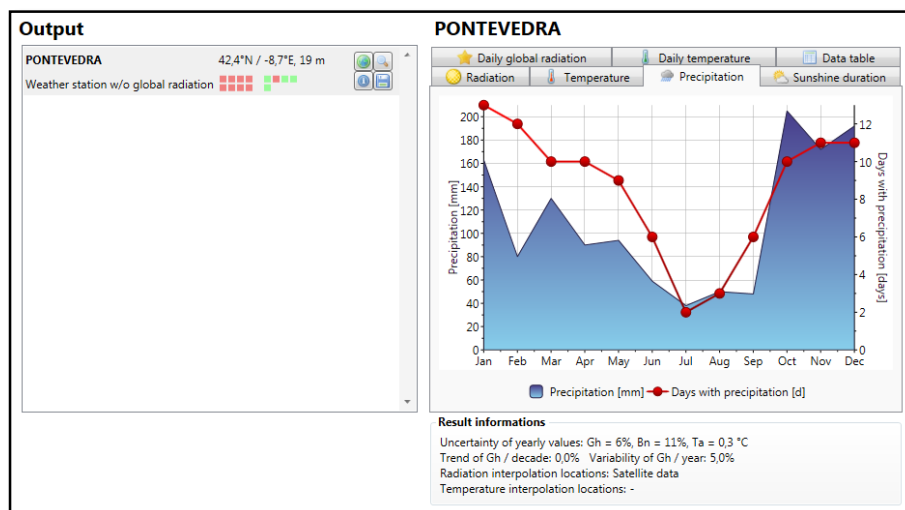
Meteonorm aplica métodos matemáticos y modelos físicos para interpolar los valores de las estaciones meteorológicas más cercanas al emplazamiento que se le indique. Los modelos de interpolación sofisticados de *Meteonorm* permiten un cálculo fiable de la radiación solar, la temperatura y otros parámetros en cualquier sitio en el mundo [23].

Se puede definir y modificar la ubicación de la construcción para estudiar la radiación solar (Figura 4-29).

Figura 4-29: Modificación de datos Meteonorm

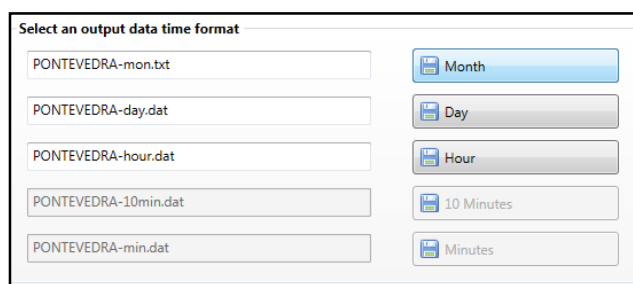
Hay tres clases principales de datos meteorológicos: "típico" (lugar durante un período arbitrario de tiempo) de uso frecuente para las condiciones de diseño y rendimiento durante la vida útil de un edificio, "actual" (lugar específico durante un período específico de tiempo) utilizados para la calibración de las facturas de energía, y "futuro" el tiempo en los datos utilizados para el control adaptativo de un edificio [3].

El propio software nos proporciona gráficas de los datos a exportar como se muestra en la Figura 4-30.

Figura 4-30: Datos obtenidos en *Meteonom*

Incluye 36 formatos diferentes de exportación predefinidos, que cubren la mayor parte del software de simulación reconocidos en aplicaciones de energía solar y diseño de edificios, incluyendo TMY2 y 3, EPW, *TRNSYS* así como los formatos de salida para TRY (años de referencia en Alemania), Polysun, TSOL / PVSOL, PVSyst y PHPP. Todos los formatos de exportación disponibles para los valores horarios y mensuales.

Para finalizar, permite exportar datos con diferentes frecuencias de medición como se observa en la Figura 4-31, desde meses a minutos.

Figura 4-31: Exportar datos *Meteonom*

4.1.4 Energy plus

EnergyPlus es un programa de simulación térmica y energética de edificios desarrollado por DOE (*Department of Energy*, Estados Unidos) con el que se pueden hacer estudios de demanda y consumo energético [4].

Para descargar este software, se empleará la página web propia del programa, obteniéndolo este de manera gratuita [4].

El uso que se le dará a este software en este TFG será para la creación de archivos meteorológicos que se incorporarán más tarde a *Trnsys*. Se empleará el programa *WEATHER.exe*, que una vez instalado el software *EnergyPlus* se encontrará en la siguiente ruta:

C:/EnergyPlus V8-0-0/PreProcess/WeatherConverter

4.2 Estación meteorológica usada

Se ubicará dicha estación en la terraza del Centro Universitario de la Defensa de la Escuela Naval Militar (Figura 4-32).



Figura 4-32: Ubicación de la estación meteorológica en la ENM [8]

4.2.1 Descripción de los elementos y montaje

Los elementos de los que consta la estación meteorológica utilizada son [24]:

4.2.1.1 Arduino UNO

Arduino (Figura 4-33) es una plataforma de hardware libre, que permite el uso de la electrónica de manera más sencilla. Obtiene información del entorno por medio de sus entradas analógicas y digitales. Realiza el control de motores, luces y otros sistemas. Se programa con un lenguaje de programación propio Arduino. Este software se puede descargar de manera gratuita en su página web, al igual que los drivers para cargar los códigos desde un ordenador.

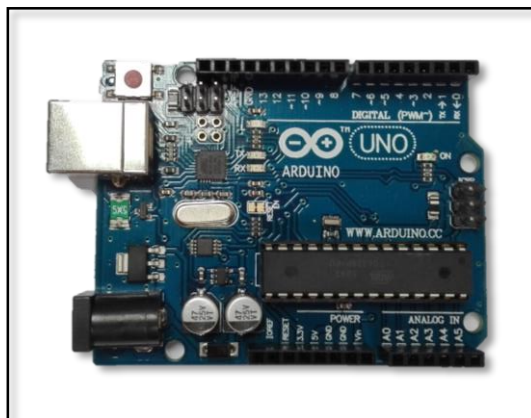


Figura 4-33: Arduino UNO

4.2.1.2 SparkFun Sunny Buddy

Red Board utilizada para proporcionar alimentación al sistema (Figura 4-34). Proporciona la capacidad de obtener la máxima potencia a partir de una placa solar fotovoltaica. Esta energía la almacena en una batería para entregársela al Arduino siempre que este la necesite. La placa fotovoltaica cargará la batería mientras no se encuentre en funcionamiento la estación, y trabajará en paralelo con la carga disponible de la batería cuando se requiera corriente para el funcionamiento de la misma. Su instalación es sencilla, únicamente hay que conectar la placa fotovoltaica y la batería

recargable a la *Red Board* y comprobar que recibe corriente. Por defecto, viene establecida una corriente máxima de 450mA con una entrada mínima de 6V.

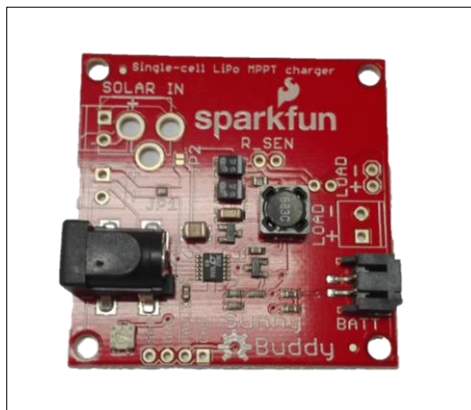


Figura 4-34: SparkFun Sunny Buddy

4.2.1.3 SparkFun Electric Imp Shield

Permite una conexión fácil e integrada de diferentes sistemas hardware y sistemas de internet. Consiste en una plataforma (Figura 4-35) que facilita la conectividad, ya que a través de una red Wi-Fi consigue la conexión del Arduino a una base de datos en internet, almacenando los datos y pudiendo visualizarlos desde cualquier ordenador. Debido a que el *Electric Imp* puede tomar hasta 400mA de la línea de 3.3V, este *shield* tiene su propio regulador de voltaje que es alimentado desde la línea de 5V del Arduino.



Figura 4-35: SparkFun Electric Imp Shield

4.2.1.4 SparkFun Weather Shield

El *Weather Shield* (Figura 4-36) es un *shield* para Arduino que permite medir la presión barométrica, la humedad relativa, la luminosidad y la temperatura. Permite conectar a este *shield* sensores de velocidad y dirección del viento, de lluvia, GPS, luminosidad o humedad. El *Weather Shield* utiliza el sensor de humedad y temperatura HTU21D, el sensor de presión barométrica MPL3115A2, los sensores de luz ALS-PT19 y se basa en las librerías para Arduino HTU21D y MPL3115A2. Cada *shield* viene con dos espacios libres para conectores RJ11 para añadir los sensores de lluvia y de viento

y un conector de 6 pines para el GPS. Puede trabajar desde 3.3V hasta 16V y está dotado de reguladores de voltaje y de traductores de señales.



Figura 4-36: SparkFun Weather Shield

4.2.1.5 RJ11 6-Pin Connector

Conector tipo teléfono empleado para proporcionar una señal de entrada a la *SparkFun Weather Shield*. En nuestra estación se emplearán dos conectores RJ11 (Figura 4-37) uno de ellos para recibir señales de viento y el otro señales de lluvia.

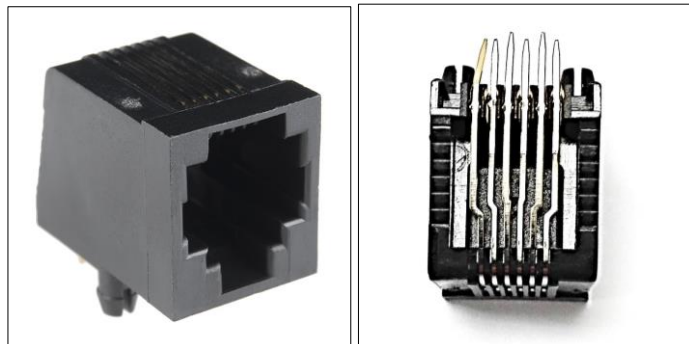


Figura 4-37: RJ11 6-Pin Connector

4.2.1.6 Arduino Stackable Header Kit - R3

Estas cabeceras mostradas en la Figura 4-38, permiten disponer los diferentes elementos en una estructura tipo torre, dejando espacio suficiente entre ellos para comprobar, limpiar e intercambiar elementos como la *Electric Imp*. Se emplean cabeceras de 10 y de 6 pines según sea necesario por diseño del Arduino.

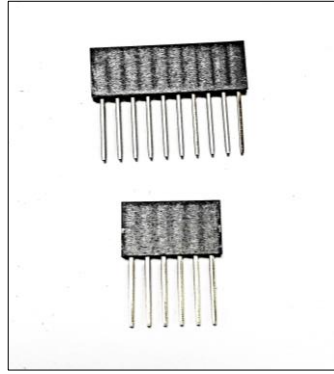


Figura 4-38: Arduino Stackable Header Kit - R3

4.2.1.7 Electric Imp

Electric Imp es una tarjeta de desarrollo que en apariencia es idéntica a una tarjeta de memoria SD (Figura 4-39). *Electric Imp* no es una tarjeta de memoria SD, no dispone de ningún tipo de memoria. Se utiliza el mismo formato que las tarjetas de memoria SD por razones de coste. En su interior dispone de conectividad por Wi-Fi, un procesador Cortex-M3 que tiene un consumo bajo de energía y un buen rendimiento para su tamaño. Los seis conectores de la tarjeta pueden utilizarse como UART, I2C, SPI, entrada y salida analógica, PWMs, GPIOs... y todo ello seleccionable por software. Se podría estar compilando y ejecutando código sobre un *Electric Imp* que se encuentre en cualquier parte del mundo con una única conexión a internet.



Figura 4-39: Electric Imp

4.2.1.8 Solar Cell Large

Placa solar (Figura 4-40) para cargar la batería y proporcionar alimentación al Arduino. Está diseñada para proporcionar 8V y 310mA en circuito abierto.



Figura 4-40: Solar Cell Large

4.2.1.9 Polymer Lithium Ion Battery - 6Ah

Batería recargable por medio de la placa solar usada como alimentación del Arduino (Figura 4-41). Está formada por tres células, que proporcionan una corriente nominal de 3.7V.



Figura 4-41: Polymer Lithium Ion Battery

4.2.1.10 Elementos de medición

La estación meteorológica propia estará formada por diversos elementos de medición, que proporcionaran el registro de datos en la *Electric Imp*, conectando los cables telefónicos a la *SparkFun Weather Shield*. Se compondrá de un anemómetro (Figura 2-8), una veleta para indicar la dirección del viento (Figura 4-42) y un pluviómetro (Figura 2-10).



Figura 4-42: Veleta de viento

4.2.1.11 Soporte y estructura

La disposición de los elementos de medición debe ser estudiada para evitar malas mediciones debido a interferencias. Se sigue el diseño de la empresa distribuidora (*SparkFun*) y se emplean los soportes tanto verticales como horizontales mostrados en las Figura 4-43, Figura 4-44 y Figura 4-45:



Figura 4-43: Soporte superior para anemómetro y veleta



Figura 4-44: Soporte medio para pluviómetro



Figura 4-45: Barra soporte principal

Una vez montados todos los soportes queda con la siguiente disposición (Figura 4-46 y Figura 4-47):



Figura 4-46: Distribución de sensores de la estación



Figura 4-47: Distribución de sensores de la estación 2

4.2.2 Montaje del sistema

Una vez se tengan montados los sensores, es el momento de montar el Arduino. Se comienza preparando la *SparkFun Electric Imp Shield*, para lo cual se deben hacer tres pasos:

Cortar el enlace de Tx (transmisión) y Rx (recepción), cortar el enlace CD-AO para evitar interferencias en las mediciones de la dirección del viento, y crear un “jumper” entre P1 y RST.

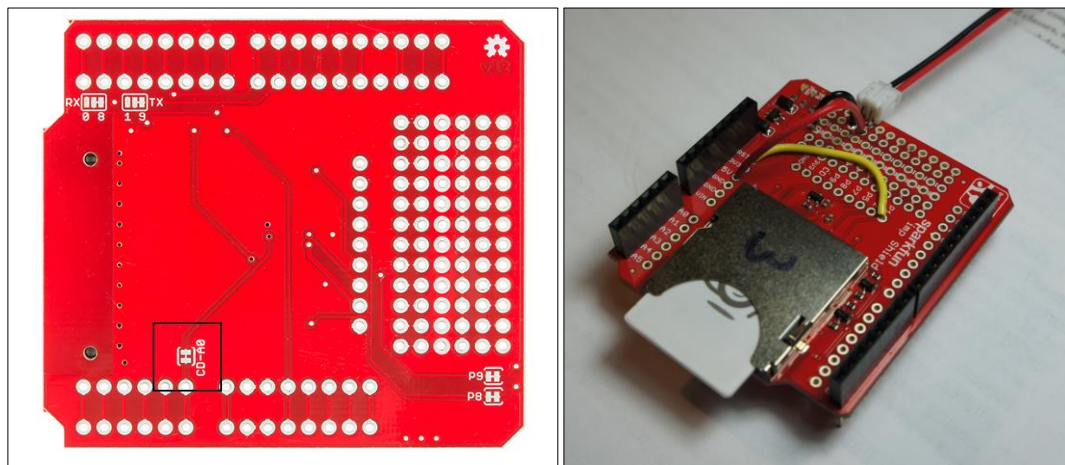


Figura 4-48: Modificación SparkFun Electric Imp Shield

Se deben poner los conectores RJ11 6-Pin Connector sobre la *SparkFun Weather Shield*, para conectar los sensores de viento y lluvia por medio de cables telefónicos. Usando los cabezales de 10 y 6 pines se debe crear una estructura de tipo torre (Figura 4-49), quedando en la parte inferior el arduino, en la parte central la *SparkFun Electric Imp Shield* y en la parte superior la *SparkFun Weather Shield*. Para conectar las diferentes capas entre sí, deben soldarse los diferentes pines con estaño como muestra la Figura 4-50, quedando de la siguiente forma:

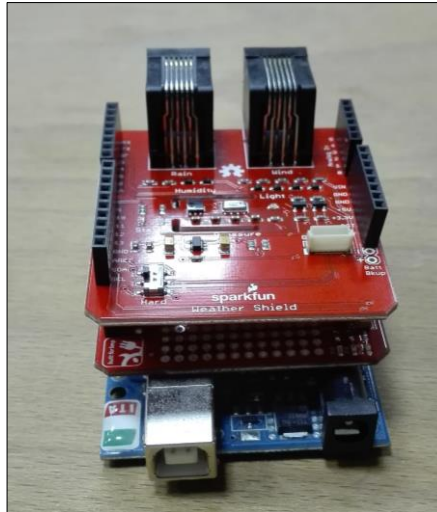


Figura 4-49: Montaje del Arduino

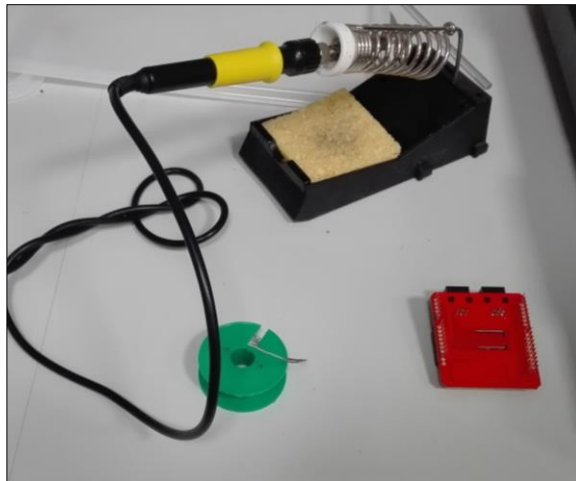


Figura 4-50: Soldadura de los pines

Para alimentar la estación meteorológica debe emplearse la *SparkFun Sunny Buddy*, conectando la *Solar Cell Large* y la *Polymer Lithium Ion Battery* como se observa en la Figura 4-51.



Figura 4-51: Alimentación de la estación meteorológica

Por último y antes de conectar todos los elementos de la estación, se debe introducir el código en el Arduino.

Los pasos a seguir son:

- Instalación del Arduino IDE: Se descarga el software gratuito ARDUINO 1.6.7 (Figura 4-52) de su página web (<https://www.arduino.cc/en/Main/Software>) y se instala en el ordenador que se usará para la programación.
- Instalación automática de los drivers que permitirán la comunicación del Arduino con el programa, necesario para introducir el código. Se abrirá la pantalla principal del software, donde se debe configurar el Arduino que se desea conectar.
- Configuración del Arduino IDE, seleccionando en primer lugar el modelo a utilizar, siendo en este caso Arduino UNO del menú “Herramientas/ Placa” del software, como muestra la Figura 4-53.
- Selección del puerto de comunicación en el menú “Herramientas/ Puerto”, el ordenador detectará el Arduino como COM3 como muestra la Figura 4-54.
- Instalación de las librerías necesarias para la estación meteorológica, utilizando el interfaz Arduino. Se instalan en el menú “Programas/ Incluir librerías/ Gestionar librerías”. Se deben instalar las librerías SparkFun HTD21D y SparkFun MPL3115A2, para permitir la conexión de los elementos de medición (Anemómetro, veleta, y pluviómetro) (Figura 4-55 y Figura 4-56).
- Se debe verificar el código usando el botón superior de la izquierda (señalado con un círculo rojo en la Figura 4-57). El programa compila el código e informa de cualquier posible error. En caso de ser el código correcto, mostrará en la parte inferior “Compilado”.
- Una vez verificado el código lo subiremos al Arduino con el botón “subir” de la parte superior izquierda del software (señalado con un círculo rojo en la Figura 4-58).
- Para comprobar que existe comunicación entre el Arduino y el programa debemos seleccionar en la parte superior derecha la lupa (señalado con un círculo rojo en la Figura 4 59), con lo que se abrirá una ventana como la mostrada en la Figura 4 59. Para comprobar la comunicación escribiremos “!” y seleccionaremos Enviar, debiendo recibir como respuesta la información de los sensores.

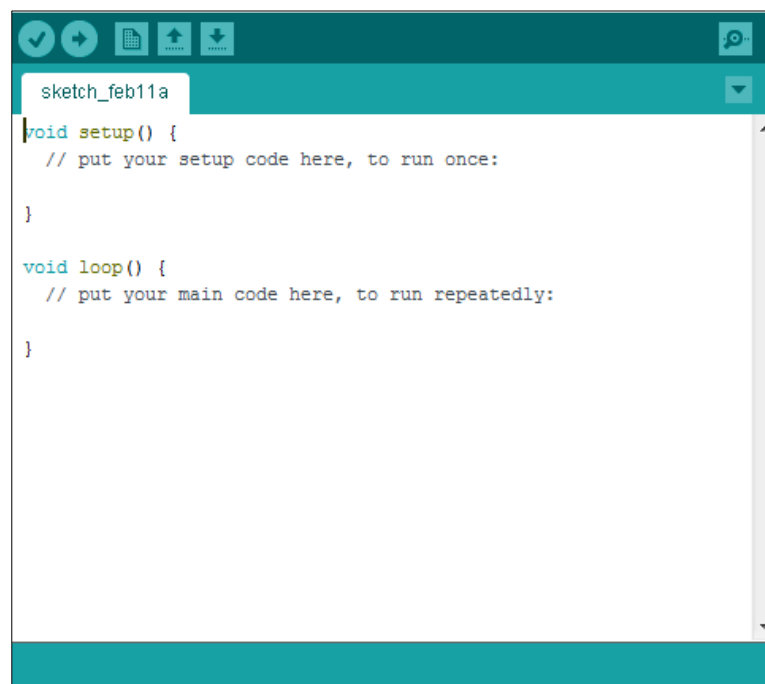


Figura 4-52: Software Arduino UNO

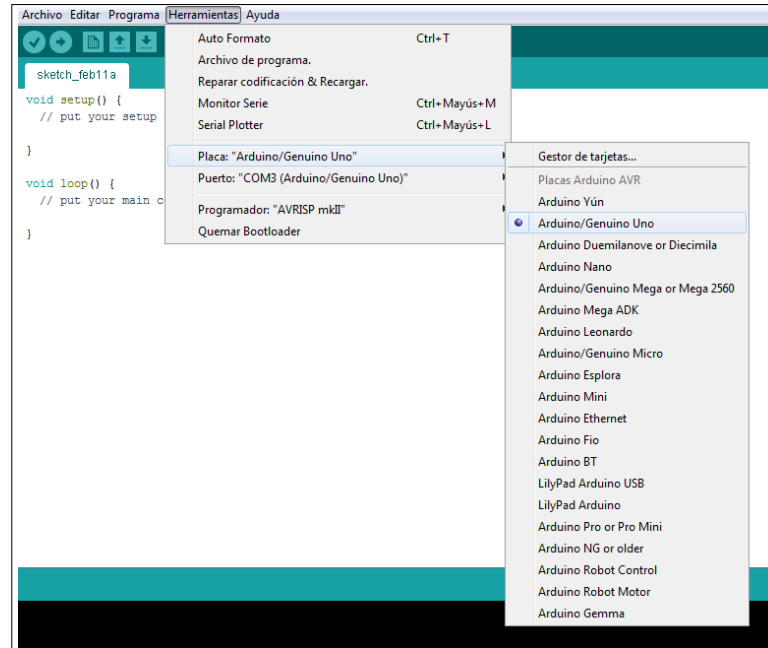


Figura 4-53: Selección de placa Arduino UNO en software

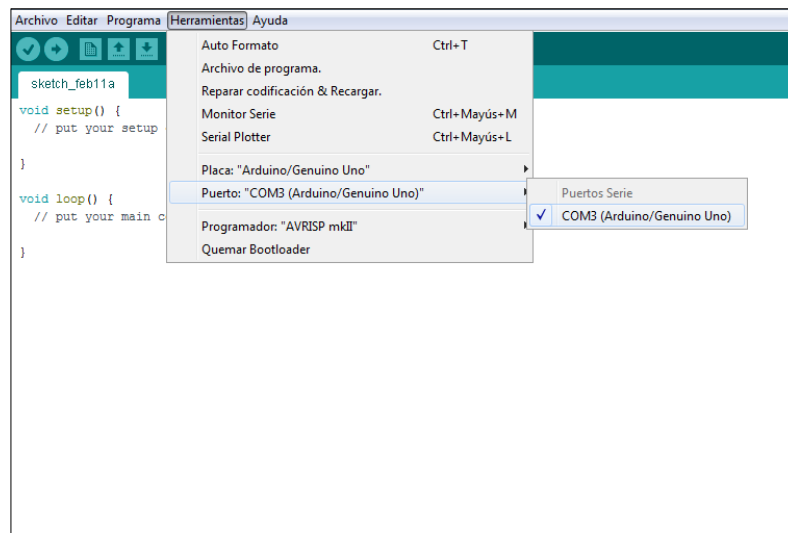


Figura 4-54: Selección de puerto COM3 en software

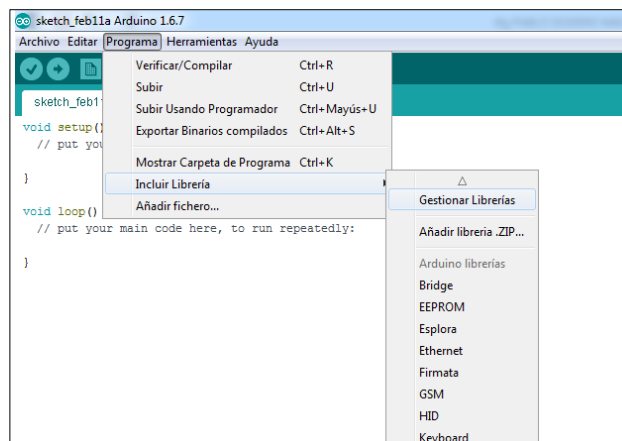


Figura 4-55: Gestionar librerías Arduino

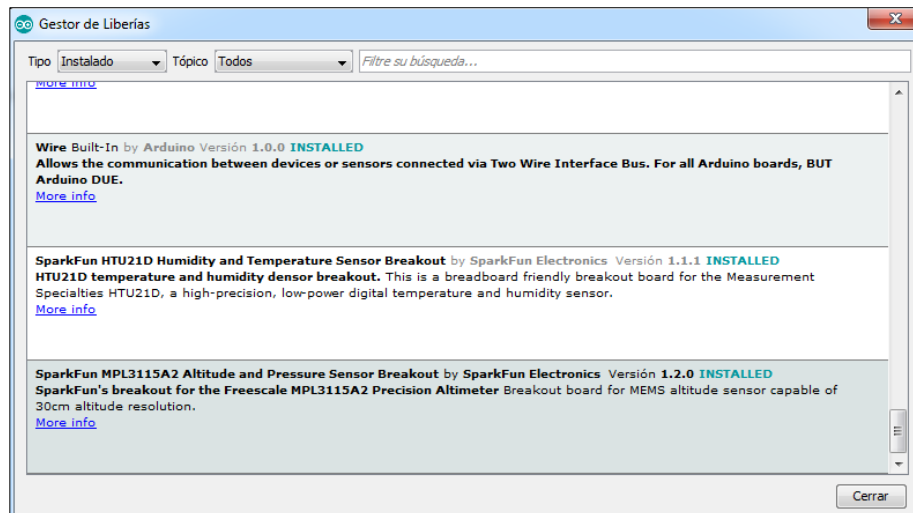


Figura 4-56: Librerías Arduino

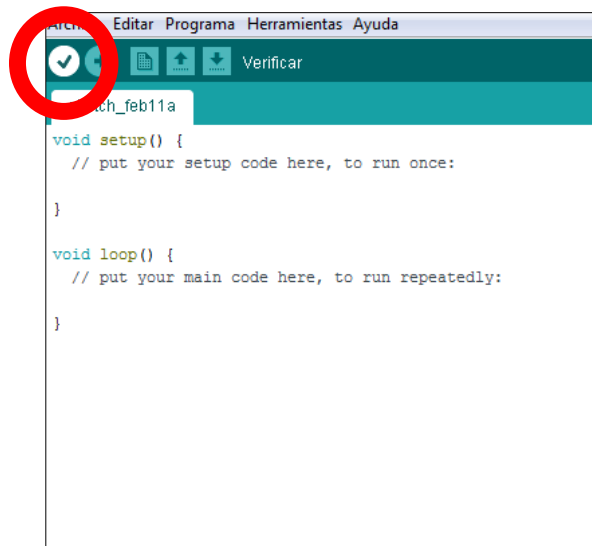


Figura 4-57: Verificar código Arduino

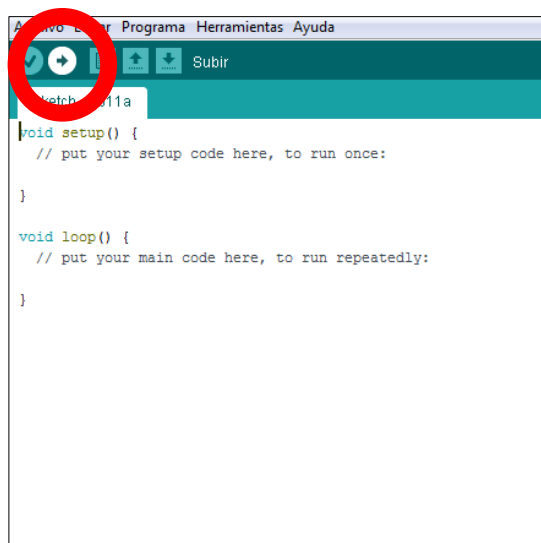


Figura 4-58: Subir código Arduino

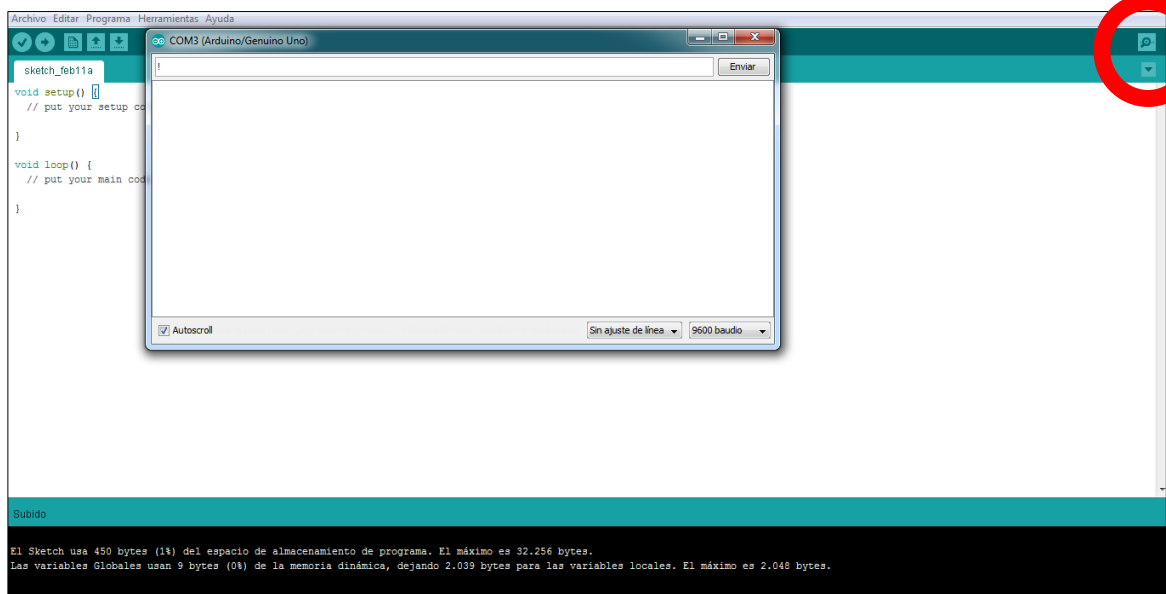


Figura 4-59: Comprobación de comunicación con el Arduino

Para continuar con la configuración y programación del sistema se debe activar la tarjeta *Electric Imp* para permitir la comunicación inalámbrica entre el Arduino y la página web de internet. Se activa dicha tarjeta con el teléfono móvil, para lo cual debe descargarse la aplicación gratuita “*electric imp*”. Esta tarjeta se comunica con la página web a través de una red Wi-Fi como muestra la Figura 4-60.



Figura 4-60: Esquema de funcionamiento de la tarjeta *Electric Imp*

Se debe crear un usuario en el siguiente enlace: <https://ide.electricimp.com/login>. Para vincular la tarjeta al usuario, es necesario con ayuda de la aplicación, generar en el teléfono móvil una serie de señales luminosas y acercar el sensor de la tarjeta (señalada con un círculo rojo en la Figura 4-61) a la pantalla del teléfono, tal y como se muestra en la Figura 4-62.



Figura 4-61: Tarjeta *Electric Imp*



Figura 4-62: Vincular tarjeta *Electric Imp* con página web[24]

Una vez asociada la tarjeta, se selecciona la red Wi-Fi que se empleará para establecer la comunicación. La red Wi-Fi utilizada es la del Centro Universitario de la Defensa, puesto que su repetidor se encuentra en un cuarto con línea de visión directa con el lugar de su instalación. Esto permite una comunicación inalámbrica de la *Electric Imp* con la página web *Wunderground* que procesa y almacena los datos de la estación en la página web como se puede observar en la Figura 4-63 (<http://www.wunderground.com/personal-weather-station/dashboard?ID=IGALICIA56#history>).



Figura 4-63: Datos estación en *Wunderground*

La programación se realizó mediante una interfaz WEB llamado *Imp IDE* (Figura 4-64). Dentro del interfaz, se copió y pegó el código facilitado en el tutorial en sus ventanas correspondientes. La ventana *device* es el código ejecutado por la tarjeta, recogida de datos de la estación, y la ventana *agent* el ejecutado remotamente en la nube, envío de datos a *Wunderground*.

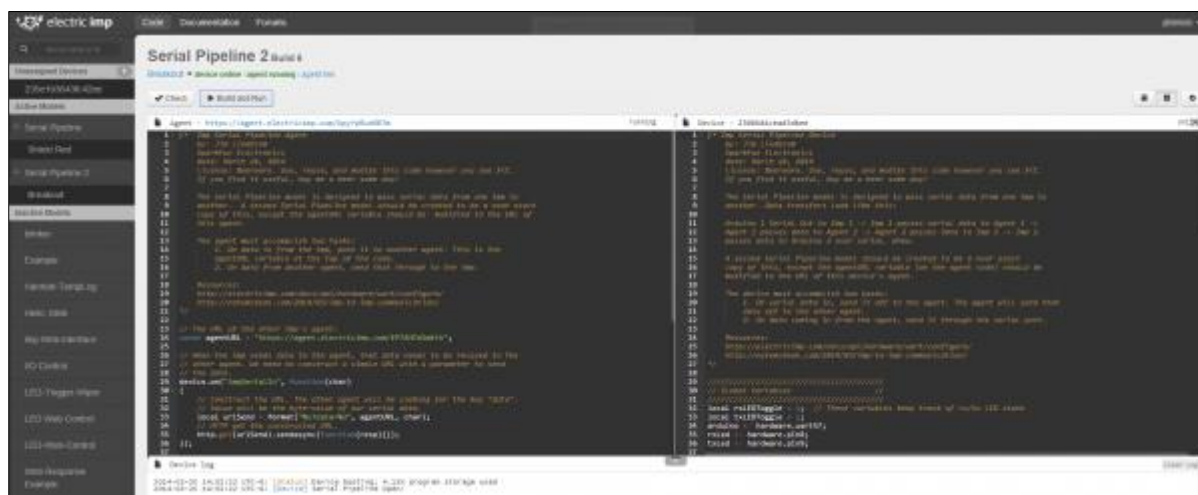


Figura 4-64: Interfaz WEB Imp IDE

Al ejecutar el código, el *shield* que utilizaba la tarjeta era incapaz de acceder al puerto serie. Al cabo de unos instantes, arrojaba un error “*Timeout Error*” producido por tal incapacidad. Se postearon consultas en los siguientes foros de usuarios:

<https://forums.electricimp.com/discussion/4137/imp-serial-timeout-error#latest>

<https://forum.sparkfun.com/viewtopic.php?f=14&t=43006>

El problema resultó ser un montaje incompleto. Se cortaron las pistas indicadas en el tutorial de su montaje (<https://learn.sparkfun.com/tutorials/weather-station-wirelessly-connected-to-wunderground>). En un principio se pensó en pistas dañadas debido a un corte demasiado impreciso.

En el momento de comprobar las conexiones, se observó que los puntos RX y TX estaban aislados. Por ello, releendo las instrucciones, se detectó un paso soslayado: la unión mediante punto de soldadura de los pines RX y TX con su respectivo pin vecino. De esta forma, la tarjeta *Electric Imp* fue capaz de acceder a los datos proporcionados por el *shield* meteorológico y enviarlos a *Wunderground*.

4.2.2.1 Instalación de la estación meteorológica en su ubicación

Una vez programado el Arduino y teniendo la estación meteorológica lista para funcionar se tuvo que crear un soporte para que pudiera permanecer en el exterior y al intemperie. Se comienza con una caja de plástico, a la que se le realizan cuatro agujeros. Se utiliza un taladro con corona (Figura 4-65) para efectuar el agujero en la tapa superior, el cual será para permitir a la placa del Arduino recibir radiación solar, tapando el agujero con una tapa circular de plástico transparente (Figura 4-66). Para efectuar los agujeros laterales, en los que se colocaran dos codos de tubería de PVC para permitir la circulación de aire e impedir la entrada de lluvia, se utiliza un taladro vertical y una escofina para conseguir obtener el agujero de tamaño adecuado para la tubería (43mm). Por último se utiliza el taladro vertical (Figura 4-67) para realizar el último de los agujeros, en el que colocaremos un prensa estopa para la entrada de los cables de la placa solar y los cables telefónicos de los sensores.



Figura 4-65: Corona del taladro

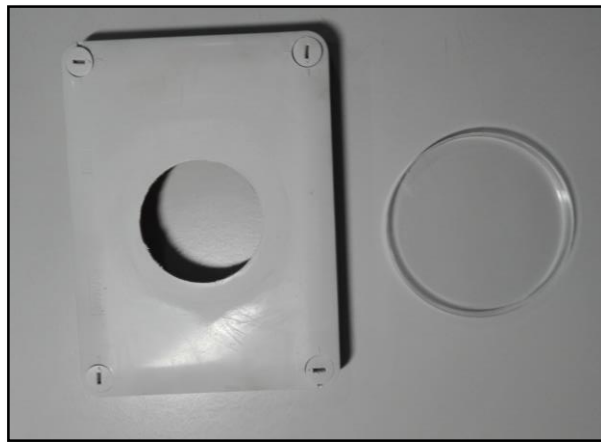


Figura 4-66: Agujero superior para la entrada de radiación solar



Figura 4-67: Taladro vertical

Una vez realizado el soporte del Arduino, se deben introducir todos los elementos de la estación (Figura 4-68) meteorológica en la caja, para lo cual se emplea una pequeña estructura para dividir la caja en dos espacios, ubicando en el espacio interior el sistema de alimentación y en la superior el Arduino con las entradas de sensores (Figura 4-69).

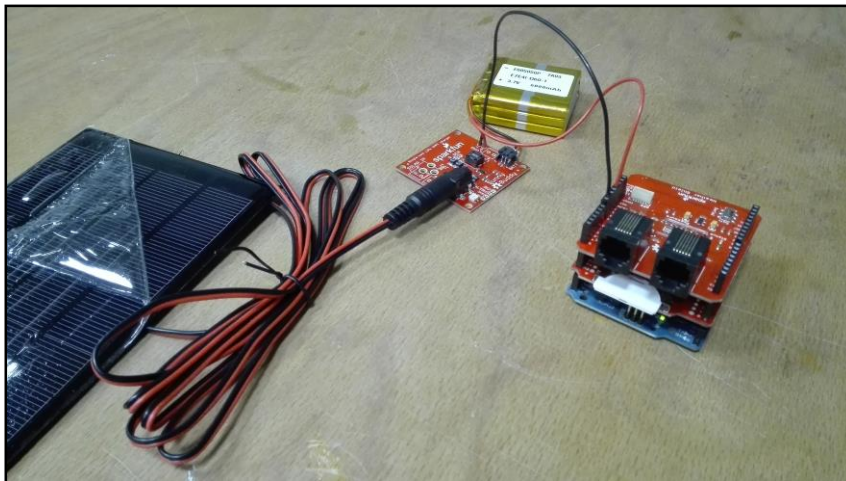


Figura 4-68: Componentes de la estación meteorológica

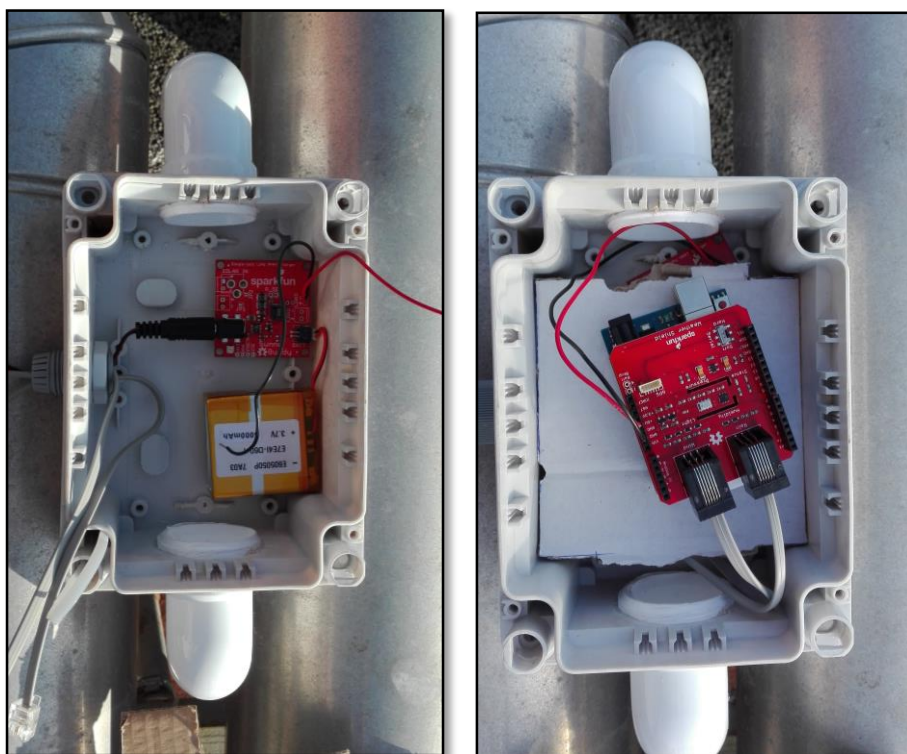


Figura 4-69: Distribución de los componentes en el interior de la caja

Se debe tener especial atención a la colocación del sensor de luz centrado con la tapa de plástico transparente para poder recibir radiación solar como se observa en la Figura 4-70.



Figura 4-70: Caja para el Arduino

Por último se debe colocar en su ubicación para probar los datos que recibe, colocándolo como muestra la Figura 4-71, sobre los tubos del sistema de refrigeración del CUD, lo que produce alteraciones en la medición de las temperaturas debido a la colocación de los codos que deben recircular el aire de la caja, en posición vertical, y de esta manera afecta directamente la temperatura de los tubos en la medición real.



Figura 4-71: Primera ubicación de la estación meteorológica

Comprobadas estas mediciones incorrectas, muy superiores comparándolas con las estaciones cercanas, se decide variar la ubicación de la caja alejando esta de las tuberías (Figura 4-72). Se coloca elevada 40 centímetros del suelo para permitir la entrada de aire.



Figura 4-72: Segunda ubicación de la estación meteorológica

4.2.3 Código usado

Se emplea el código obtenido en los tutoriales de SparkFun para el proyecto de Weather Station [24]. Éste código se adjunta en los anexos siguientes: Anexo V: Código Arduino, Anexo VI: Código Device Imp IDE y Anexo VII: Código Agent.

4.3 MeteoGalicia: datos temperatura del agua de mar

Para realizar las simulaciones térmicas en buques, se debe considerar la temperatura del agua de mar, ya que como se concluirá en el apartado “Influencia de la temperatura del agua de mar en la simulación” afecta de manera considerable a las temperaturas internas de la obra viva del buque. Por ello, obtenemos datos de MeteoGalicia para posteriormente poder analizar la variación entre la temperatura medida in situ y la proporcionada por los servicios meteorológicos de la Junta de Galicia.

Se puede obtener gran cantidad de información en la página web, así como predicciones, observaciones o modelos numéricos. Se utilizan para este proyecto los datos de temperatura del agua de mar. Se estudian las plataformas disponibles en MeteoGalicia que puedan proporcionar datos de la temperatura de mar, buscando en el apartado “Observación” y posteriormente “Red Oceanográfica”. Se decide utilizar la boya oceanográfica de uno de los pilares del puente de Rande, por ser la más cercana a la ubicación en la que se realiza la simulación, distanciando las dos ubicaciones una distancia de 12.22 kilómetros como se puede observar en la Figura 4-73 [6, 8].

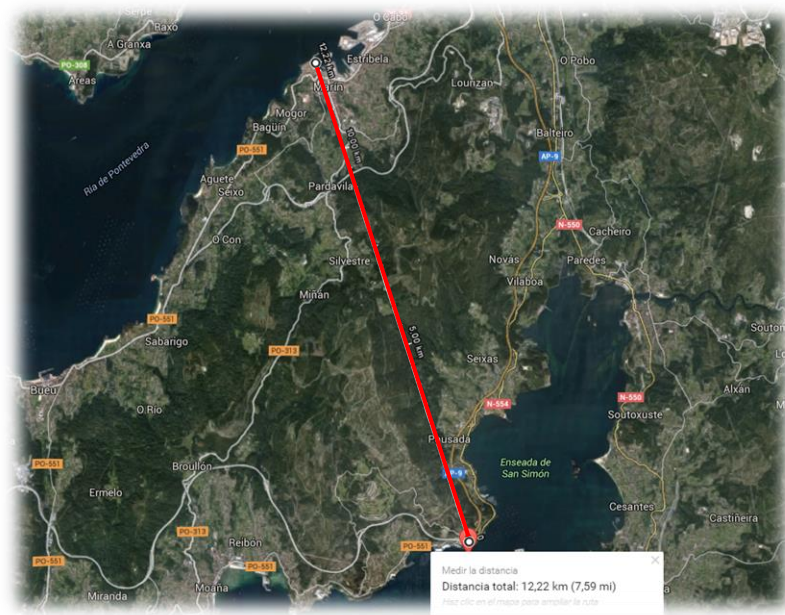


Figura 4-73: Distancia entre Puente de Rande y patrullero Tabarca [8]

La plataforma del puente de Rande (ubicada en $42^{\circ} 17,19'N$ de latitud y $008^{\circ} 39,6'W$ de longitud) dispone de una sonda de temperatura, proporcionando datos a diferentes profundidades, siendo las de superficie y fondo las más estudiadas. Proporciona datos en diferentes formatos para adaptarse a las necesidades de los usuarios (Figura 4-74).



Figura 4-74: Datos de la plataforma del puente de Rande [6]

Se utilizarán los datos de tablas y gráficos de la Figura 4-75, utilizando las profundidades de superficie y fondo para posteriormente comparar con los datos que mediremos en la ubicación del Tabarca.



Figura 4-75: Datos de temperatura del agua de mar en tablas y gráficos [6]

4.4 Construcción de archivos meteorológicos (.epw)

4.4.1 Variables utilizadas

Para la construcción de archivos meteorológicos, los cuales se emplearán para realizar simulaciones térmicas con mayor exactitud, se definen como variables de estudio las siguientes:

4.4.1.1 Temperatura

Se define temperatura como la cantidad de energía interna que tiene un sistema termodinámico de acuerdo al movimiento de sus partículas, bien sea, un objeto, cuerpo o ambiente. Como toda magnitud física, la temperatura tiene asociadas unas unidades de medida, diferentes en función de la escala que elegimos [25, 26]:

Escala Celsius (° C): Consiste en una división regular en 100 intervalos, donde el 0 corresponde al punto de congelación del agua y el 100 al punto de ebullición del mismo. Se expresa en grados centígrados y es la que utilizamos habitualmente.

Escala Fahrenheit (° F): El termómetro se gradúa entre 32 ° F (correspondiente a los 0 ° C) y 212 ° F (correspondientes a los 100 ° C).

Escala Kelvin (K): Es una escala que no tiene valores negativos de la temperatura y su cero se sitúa en el estado en el que las partículas que forman un material no se mueven. El punto de ebullición del agua corresponde a 373 K y el de congelación a 273 K.

En las simulaciones térmicas se introducirán la temperatura del agua de mar, obtenida con sensores propios o con fuentes externas como MeteoGalicia y la temperatura ambiente, que es la temperatura del aire en la ubicación en la que se mide, y se obtendrán como resultados las temperaturas internas de las zonas térmicas que se creen.

Según la Organización Meteorológica Mundial (OMM), la temperatura del aire es la temperatura que mide un termómetro protegido del Sol y de otras fuentes de radiación, y de las precipitaciones, pero bien expuesto al aire [5].

Se debe definir la temperatura de punto de rocío y la temperatura de bulbo seco, ya que son los datos de temperatura que medirá la estación meteorológica.

Temperatura de punto de rocío (T_d): Es la temperatura a la que se satura el aire húmedo (humedad relativa 100%), es decir cuando comienza a condensarse el vapor de agua debido al enfriamiento, manteniendo la presión y la razón de humedad constante (Ecuación 4-1).

Ecuación 4-1: Temperatura del punto de rocío

$$P_r = \sqrt[8]{\frac{H}{100} [112 + (0.9T)] + (0.1T) - 112}$$

P_r = Punto de rocío

T = temperatura en ° Celsius

H = humedad relativa

Temperatura de bulbo seco (T): Es la verdadera temperatura del aire húmedo que marca un termómetro común, se le denomina de manera frecuente temperatura del aire.

4.4.1.2 Dirección e intensidad del viento

Se denomina viento al desplazamiento de masas de aire, por diferencias de presión y/o temperatura. La radiación solar calienta la tierra, y esta calienta el aire que está en contacto con su superficie. Al calentarse el aire se dilata, por lo que su densidad disminuye y asciende hasta que su temperatura se iguala con la del aire circundante [4, 27, 28].

Para medir la dirección se utilizan veletas y se nombran los vientos de diferente manera dependiendo del ámbito de aplicación, por norma general, se indica lugar de procedencia y se nombra con los puntos cardinales por medio de la rosa de los vientos de la Figura 4-76 (N, S, E, W y sus derivaciones) o con nomenclatura sexagesimal.

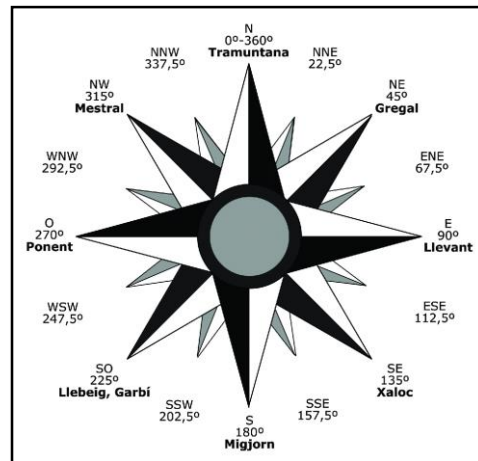


Figura 4-76: Rosa de los vientos

Para medir la intensidad se emplean anemómetros normalmente de cazoletas. Según la intensidad se clasifican con diferentes nombres según la Escala de Beaufort (Anexo I: Escala de Beaufort). EN determinados ámbitos como en la navegación y vuelo se mide la intensidad en nudos ($1 \text{ nudo (kt)} = 1 \text{ milla náutica por hora} = 1852 \text{ m/h}$).

4.4.1.3 Presión atmosférica

La presión atmosférica es la fuerza por unidad de área que ejerce el aire sobre la superficie terrestre, se mide en Pascales en el Sistema Internacional.

La presión atmosférica en un punto que coincide numéricamente con el peso de una columna estática de aire de sección recta unitaria que se extiende desde ese punto hasta el límite superior de la atmósfera. Como la densidad del aire disminuye conforme aumenta la altura, no se puede calcular ese peso a menos que seamos capaces de expresar la variación de la densidad del aire (ρ) en función de la altitud o de la presión [29].

Existen muy diversas unidades de medida de la presión atmosférica. Las más comunes son: atmósferas, mm de mercurio, pascales, hectopascales y milibares. La conversión entre unas y otras puede realizarse teniendo en cuenta que: $1 \text{ atmósfera} = 760 \text{ mmHg} = 101300 \text{ N/m}^2 \text{ (o Pa)} = 1013 \text{ mb (o hPa)}$.

4.4.1.4 Humedad relativa

La humedad relativa (Ecuación 4-2), HR [%], es la proporción de vapor de agua real en el aire comparada con la cantidad de vapor de agua necesaria para la saturación a la temperatura correspondiente. Indica que tan cerca está el aire de la saturación [30]. Se mide en porcentaje entre 0 y 100, donde el 0% significa aire seco y 100% aire saturado:

Ecuación 4-2: Humedad relativa

$$HR = \frac{e}{e_s} \cong \frac{r}{r_s}$$

Donde,

e: Es la razón entre la masa de vapor de agua y la masa de aire seco.

e_s: Es la razón entre la masa de vapor de agua y la masa de aire seco cuando el aire está saturado.

r: Presión parcial de vapor de agua.

r_s: Presión parcial de vapor de agua cuando el aire está saturado.

Nótese que la humedad relativa suele darse siempre en porcentaje (e.g. HR=60%, o más comúnmente escrito como 60% HR, en lugar de dar el cociente unitario, HR =0,6). La humedad relativa en superficie varía mucho espacial y temporalmente, entre un 10% HR en los desiertos más secos, hasta un 100% en situación de niebla persistente.

4.4.1.5 Radiación global horizontal

La radiación global se define como la radiación solar recibida de un ángulo sólido de 2π estereorradianes sobre una superficie horizontal. La radiación global incluye la recibida directamente del disco solar y también la radiación celeste difusa dispersada al atravesar la atmósfera [31].

La radiación global horizontal se suele medir con un piranómetro, aparato que usa un lente de 180° para conducir toda la radiación disponible hacia un sensor termoeléctrico que mide la energía recibida.

4.4.2 Esquema de procedimiento de construcción de archivos

Un archivo del clima contiene datos de las variables meteorológicas explicadas en el apartado anterior, siendo estas medidas en cada intervalo de tiempo definido por el usuario. Usar archivos meteorológicos es adecuado para el proceso de simulación energética, ya que el programa utilizara para el cálculo numérico variables medidas cada hora y no promedios [30].

Para la creación de archivos meteorológicos, se necesita tener el archivo .csv con el archivo de formato .def, compartiendo ambos archivos el mismo nombre y carpeta. El archivo .csv incluye los datos de la estación, mientras que el archivo .def incluye los datos de la ubicación donde se toman los datos.

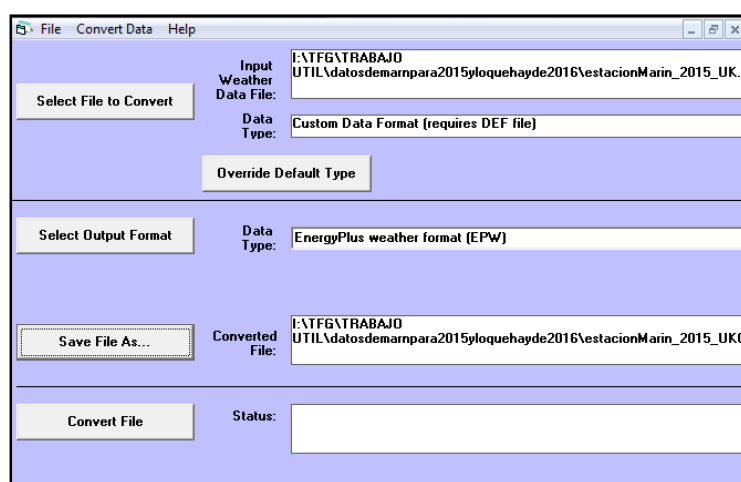


Figura 4-77: Software Weather de EnergyPlus

Se debe seleccionar el archivo .csv en *Select File to Convert*, de manera automática seleccionará el archivo .def como se muestra en la Figura 4-77. Una vez seleccionado el formato en el que se desea obtener el archivo (epw), se debe seleccionar donde guardar el archivo y hacer clic en *Convert File*, el propio programa mostrará un aviso cuando se haya realizado la conversión, y se obtendrán una serie de archivos en la carpeta deseada como muestra la Figura 4-78.

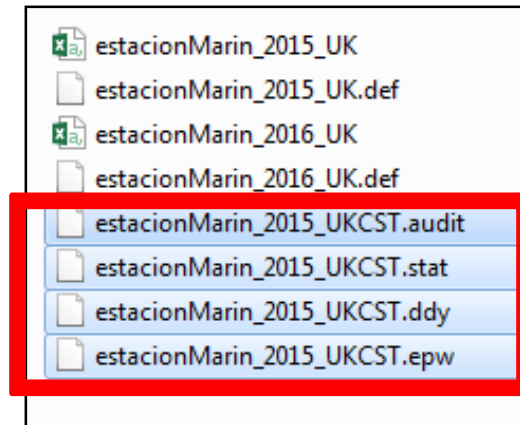


Figura 4-78: Archivos convertidos con *Weather*

El archivo meteorológico (.epw) se introducirá en *Trnsys* (Figura 4-79). Se debe realizar un cambio para obtener el funcionamiento correcto. Se debe modificar el type 56: *Weather data* (Figura 4-80) para que el type realice la lectura del archivo .epw, para lo cual, se debe seleccionar con el botón derecho y seleccionar la opción de *Replace* donde se buscará la carpeta *Weather Data Reading and Processing* y dentro de esta *Standart Format* y por último la carpeta *Energy+ Weather Files (EPW)* donde se seleccionará el type15-3.tmf. Para terminar se debe introducir el archivo creado .epw como *External File* y ejecutar la simulación.

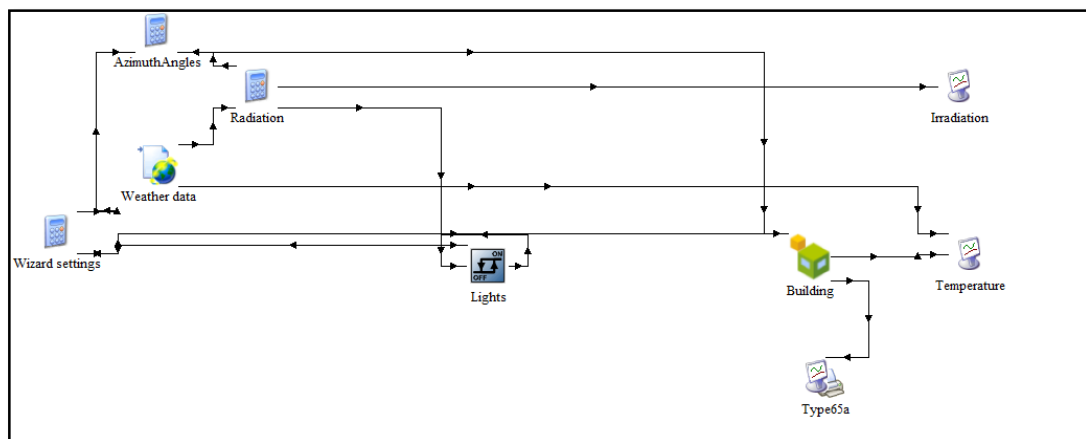


Figura 4-79: Simulación *Trnsys* para *Weather*

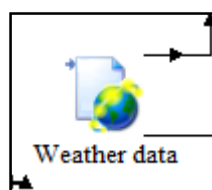


Figura 4-80: Type 56 a editar para leer archivos .epw

4.4.3 Datos meteorológicos a emplear

Para realizar las futuras simulaciones se emplearán los datos de temperatura del mar obtenidos con sensores de temperatura en la dársena de la Escuela Naval Militar. En cuanto a los datos meteorológicos a introducir en el type 15.2, se emplearán los obtenidos en Meteonorm y los datos proporcionados por el equipo de investigación GTE de la Universidad de Vigo, que obtiene los datos a partir de métodos de interpolación con kriging, empleando datos medidos en sus estaciones y trasladándolos con este método a la ubicación exacta del Tabarca en sus estancias en puerto.

4.5 Barco objeto de análisis: Patrullero Tabarca

4.5.1 Descripción general del buque

El patrullero Tabarca pertenece al Mando de las Unidades de la Fuerza de Acción Marítima de Ferrol. Perteneciente a la clase "Anaga", siendo el octavo patrullero de diez que en su origen formaban la serie.



Figura 4-81: Escudo patrullero Tabarca P28 [32]

El buque fue construido por la Empresa Nacional Bazán (actual Navantia) en San Fernando (Cádiz). Su nombre se debe a una pequeña isla de la provincia de Alicante, siendo este el primer buque de la Armada con este nombre.

Fue botado el 23 de diciembre de 1980 y entregado a la Armada el 31 de diciembre de 1981, desde esa fecha realiza labores de vigilancia costera en aguas gallegas, caladeros de pesca nacionales y Zona Económica Exclusiva. También colabora activamente con la Escuela Naval Militar para el adiestramiento de los futuros Oficiales de la Armada navegando con alumnos a bordo en la Ría de Pontevedra como se observa en la Figura 4-82, así como con unidades especiales del Ejército de Tierra y de la Armada.



Figura 4-82: Patrullero Tabarca P-28 [32]

Realiza comisiones de una semana de duración para vigilancia marítima y salidas a la mar para realizar maniobras de adiestramiento propio o de alumnos de la Escuela Naval Militar.

Actualmente solo quedan en activo tres de los nueve iniciales de la serie, Tagomago, Medas y Tabarca, como se indica en la Tabla 4-1, los cuales se prevé acaben su vida útil en los próximos años debido a la incorporación de los BAM clase Meteoro.

La dotación está compuesta por 27 personas; 1 TN Comandante, 3 Oficiales, 4 Suboficiales, 6 Cabos primeros, 4 Cabos y 9 Marineros

El buque dispone de 1 camarote para el Comandante, 1 camarote de oficiales, 1 camarote de suboficiales y sollados para la marinería. Además de las patrullas y maniobras que se realizan en la mar, el horario en puerto se divide entre la realización de mantenimientos preceptivos, alistamiento para el combate e instrucción y adiestramiento de la dotación.

La propulsión del buque está compuesta por un único motor modelo "BAZÁN MTU" que da 4.500 CV de potencia con una velocidad máxima de 15 nudos en su origen.

Además del propulsor principal, dispone de dos motores auxiliares diésel CHRISLER-BARREIROS tipo BS-36ME de 1500 rpm sobrealimentados.

En cuanto al armamento cabe destacar un montaje de 3''/50 situado en proa y una ametralladora de 20 milímetros en popa, además del dos ametralladoras móviles MG de 7.62 milímetros [32].

Tabla 4-1: Patrulleros Clase Anaga [32, 33]

<u>Nombre del buque</u>	<u>Numeral</u>	<u>Fecha botadura</u>	<u>Fecha entrega</u>
Anaga (Baja - 11/06/2010)	P-21	14 de febrero de 1980	14 de octubre de 1980
Tagomago	P-22	14 de febrero de 1980	30 de enero de 1981
Marola (Baja - 11/06/2010)	P-23	11 de abril de 1980	04 de junio de 1981

Mouro (Baja - 11/06/2010)	P-24	11 de abril de 1980	14 de julio de 1981
Grosa (Baja - 06/06/2012)	P-25	10 de diciembre de 1980	15 de septiembre de 1981
Medas	P-26	10 de diciembre de 1980	16 de octubre de 1981
Izaro (Baja - 03/12/2010)	P-27	23 de diciembre de 1980	9 de diciembre de 1981
Tabarca	P-28	23 de diciembre de 1980	31 de diciembre de 1981
Deva (Baja- 23/07/2004)	P-29	24 de noviembre de 1981	03 de junio de 1982
Bergantín (Baja - 11/06/2010)	P-30	24 de noviembre de 1981	28 de julio de 1982

4.5.2 Puerto base

Su puerto base (Figura 4-83), lugar en el que se ubica en los periodos en los que no se encuentra navegando o en periodo de reparación, es la Escuela Naval Militar de Marín (Pontevedra), donde estudian los futuros Oficiales de la Armada, razón por la cual sirve de apoyo al adiestramiento de los alumnos.



Figura 4-83: Patrullero Tabarca P-28 en su puerto base [32]

4.5.3 Aspectos constructivos del barco. Cerramientos

Los cerramientos son los componentes constructivos que delimitan los diversos espacios contenidos en un edificio. Tanto los cerramientos exteriores (suelos, muros y cubiertas) como los interiores (entresijos y muros divisorios) afectan el comportamiento térmico de los edificios. Sin embargo son los cerramientos exteriores los que, al conformar la envolvente general, proporcionan la principal barrera protectora contra los factores ambientales. Son estos cerramientos, en primera instancia, los que regulan el flujo del aire exterior, la incidencia de la radiación solar, la entrada de luz natural y la transmisión del calor (hacia el interior y hacia el exterior) [34].

La envolvente en una edificación tiene gran importancia, por lo que se debe definir de manera precisa y lo más idéntica posible a las condiciones reales para que la simulación energética sea fiable. La envolvente del edificio es la separación entre el interior y el entorno exterior de un edificio. Es de gran importancia ya que realiza la función de protección del interior, así como, control térmico. Al tratarse de un buque, se debe conseguir la máxima eficiencia energética posible con los cerramientos, para así reducir consumos empleados en climatización y mejorar su autonomía.

En el buque de estudio, se realiza el estudio de los cerramientos, obteniendo los materiales que se deben emplear para la simulación.

Se definen como aceros de construcción naval a los aceros estructurales que se utilizan en la construcción de los cascos de los buques [35].

Los aceros de construcción naval han de poseer, una resistencia suficientemente elevada como para soportar sin romperse, ni sufrir deformación permanente, a las cargas a que se ven sometidas en servicio, especialmente a tensiones de arrufo y quebranto.

Los aceros se dividen en función de sus características mecánicas:

Calidad A es acero dulce normal.

Calidad B similar al A pero un poco más resistente en las aberturas a la formación de grietas

El casco del Tabarca se trata de acero de calidad A de 7 milímetros de espesor recubierto en su interior por una capa de 5 centímetros de un aislante térmico llamado lana de roca. Para las simulaciones en Trnsys se empleará un casco de 7 centímetros analizando de esta manera un cerramiento que aisle más del exterior pudiendo así comprobar la influencia de las variables meteorológicas con cerramientos de mayor tamaño a los reales.

4.5.3.1 Geometría y volumen

Se trata de un buque de construcción totalmente soldada, de acero de calidad A naval, de 44.5 metros de eslora y 6.6 metros de manga [35].

Se define como desplazamiento de un buque el producto entre el volumen sumergido (obra viva) y el peso específico del fluido en el que flota, representando éste el volumen de agua desplazada (Principio de Arquímedes [36]).

Se debe tener en cuenta el valor de desplazamiento a emplear, ya que existen diferentes desplazamientos [37]:

- Desplazamiento en rosca: es el peso del buque sin combustible, pertrechos, víveres ni tripulantes.
- Desplazamiento estándar: es el peso del buque añadiendo, tripulación, equipos (botes, instrumentos de navegación, etc.), líquidos en circulación, víveres, munición (en los buques de guerra), agua dulce y aceite lubricante. Quedarían excluidas el combustible y el agua de reserva para las calderas.
- Desplazamiento en lastre: es el peso del buque en rosca más todo lo necesario para que pueda navegar (combustible, agua potable, provisiones y pertrechos), pero sin carga.
- Desplazamiento máximo: es el peso que alcanza cuando está sumergido hasta la línea de máxima carga (agua de mar en verano de la marca de Plimsoll. Figura 4-84).

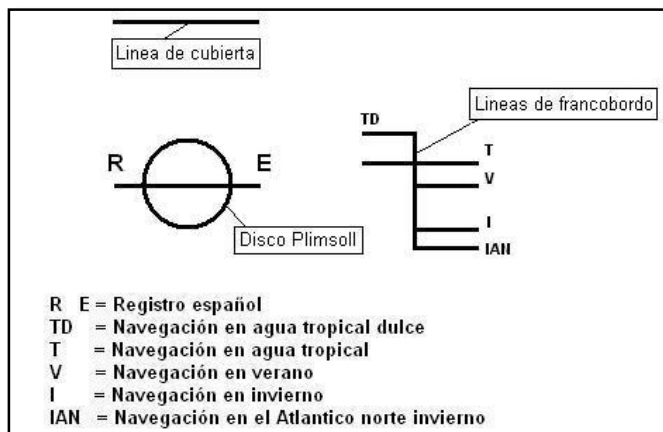


Figura 4-84: Marca de Plimsoll franco bordo [38]

El dato de desplazamiento de un buque, sino se especifica, se refiere al desplazamiento máximo.

La diferencia entre el desplazamiento máximo y el desplazamiento en rosca se denomina tonelaje de porte bruto o tonelaje de peso muerto. Así, el tonelaje de peso muerto incluye el peso de la carga, incluyendo pasajeros y tripulación, y el de los consumibles (combustible, víveres, agua potable...) mencionado más arriba.

La diferencia entre el desplazamiento máximo y el desplazamiento en lastre se conoce como «capacidad de carga», que indica el peso de la carga que es posible transportar en ese buque.

Entre la condición de desplazamiento en rosca y la de desplazamiento máximo existen infinitos valores intermedios. A medida que el buque se va sumergiendo, conforme aumenta su desplazamiento, varían toda una serie de valores, como: calado para agua de mar, calado para agua dulce, área del plano de flotación, posición del centro de flotación, volumen de carena, área de la sección maestra sumergida, momento de asiento unitario, posición del metacentro transversal, etc.

Todos estos parámetros y otros más, constituyen los atributos de la carena adrizada o derecha y son una característica de cada casco. Los astilleros entregan esta información mediante tablas y curvas denominadas curvas hidrostáticas, lo que permite conocer todos estos valores para cualquier condición de desplazamiento entre los mencionados extremos.

Es interesante destacar que para una misma condición de carga un buque tiene un único desplazamiento, pero conforme flote en agua salada o agua dulce experimentará dos calados medios diferentes, fruto de la diferencia de densidad entre una y otra (1.025 T/m^3 el agua de mar frente a 1.000 T/m^3 el agua dulce). Esta diferencia es lo que se conoce como permiso de agua dulce [39].

Al depender este dato de desplazamiento de la carga y del fluido en que se encuentre, no se trataría de un valor lo suficiente preciso para la realización de los cálculos, debido a que desde su construcción ha sido sometido a diferentes obras, en las cuales se ha variado el peso del buque, sustituyendo elementos, maquinaria o realizando reparaciones como añadir chapa en determinadas partes del casco.

Se obtienen del propio buque, los calados medidos en condiciones normales en una de sus estancias en puerto. Los calados facilitados son:

- Calado de proa: 2.60 metros.
- Calado de popa: 2.70 metros.

Con esto se obtiene un calado medio, necesario para obtener el desplazamiento en esas condiciones, entrando en las Curvas Hidrostáticas de los patrulleros Clase Anaga como se muestra en la Figura 4-85 y el Anexo III: Curvas Hidrostáticas.

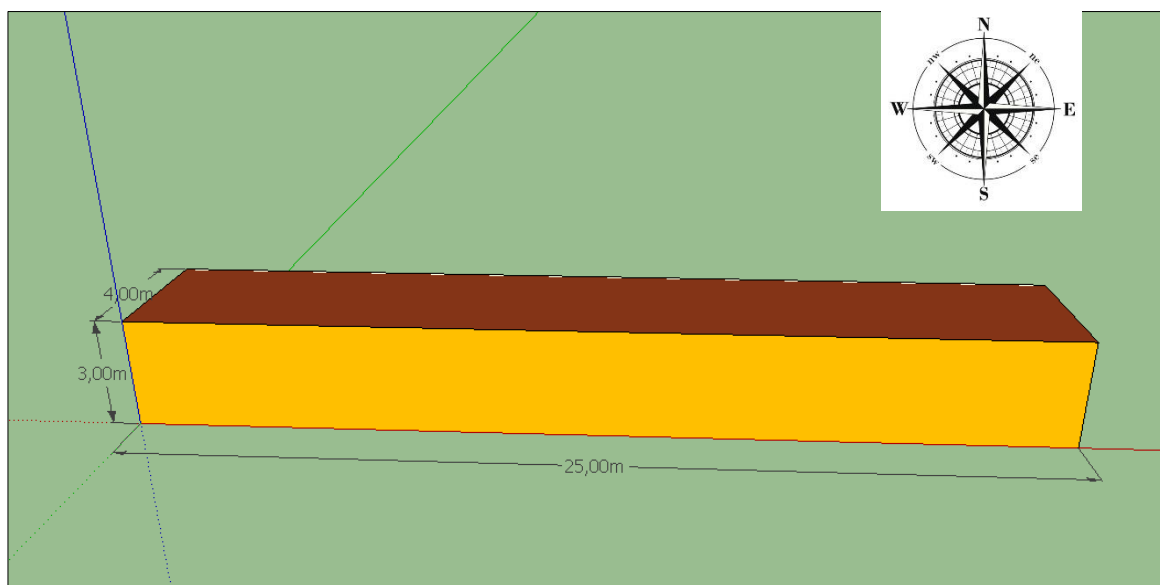


Figura 4-86. Modelo en SketchUp para una zona térmica de 300 m³

Para dotar de mayor realismo a la simulación, se utilizaron datos meteorológicos de Meteonorm para la localización de Pontevedra (Figura 4-87), punto más próximo al lugar donde tiene el puerto base el patrullero Tabarca. En la Figura 4-86 el Norte está representado por la línea de color verde del SketchUp, dato clave por la influencia de la radiación solar, marcada por la trayectoria solar. En la Figura 4-2 se muestra la ubicación concreta del volumen a simular.



Figura 4-87: Ubicación del patrullero Tabarca [8]

Para describir los parámetros característicos de la simulación, se comienza por el volumen de 300 m³ creado para realizar una aproximación al volumen de obra viva del buque (Figura 4-86), se debe exportar a *Simulation Studio* para realizar la simulación. Se debe añadir en la simulación de la Figura 4-88 un convertor de unidades (kJ a W), y diferentes types (módulos) para representar los datos (type65c) y exportarlos a un documento de texto (*System_Printer* y type 65c). Se deben configurar los *Output* que se quieren estudiar y obtener en las gráficas, para ello se establecen las conexiones entre los diferentes types.

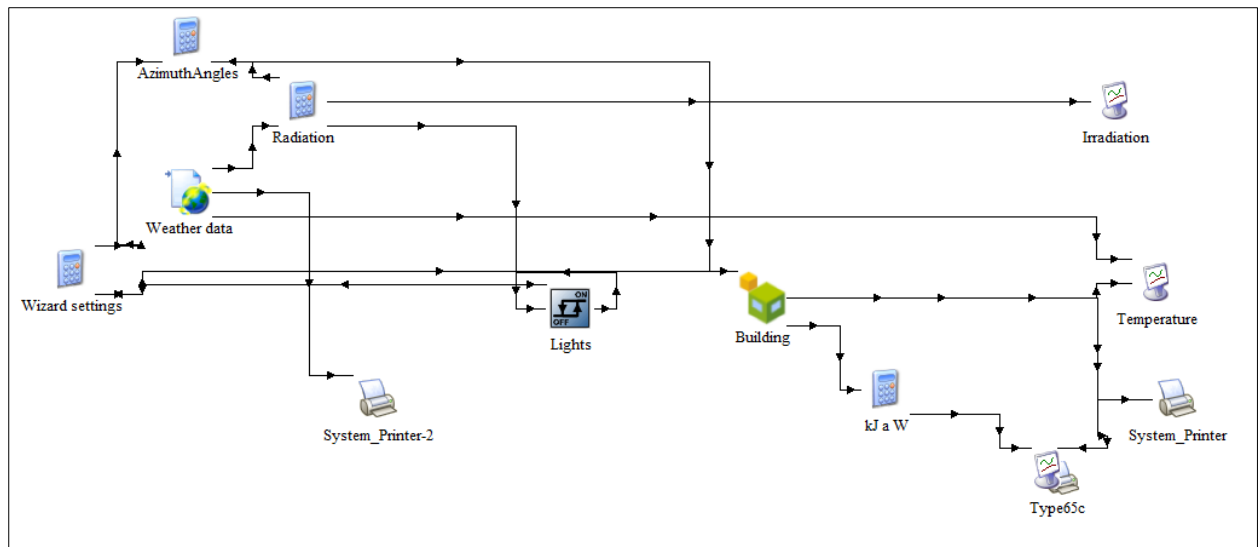


Figura 4-88: Simulación de un volumen en *Trnsys*

Se introducen los datos meteorológicos de Pontevedra. Para ello se hace doble clic sobre el type *Weather Data* y se selecciona en la parte superior del menú del type la pestaña *External Files*, donde se carga el archivo meteorológico previamente descargado de *Meteonorm* (Figura 4-89).

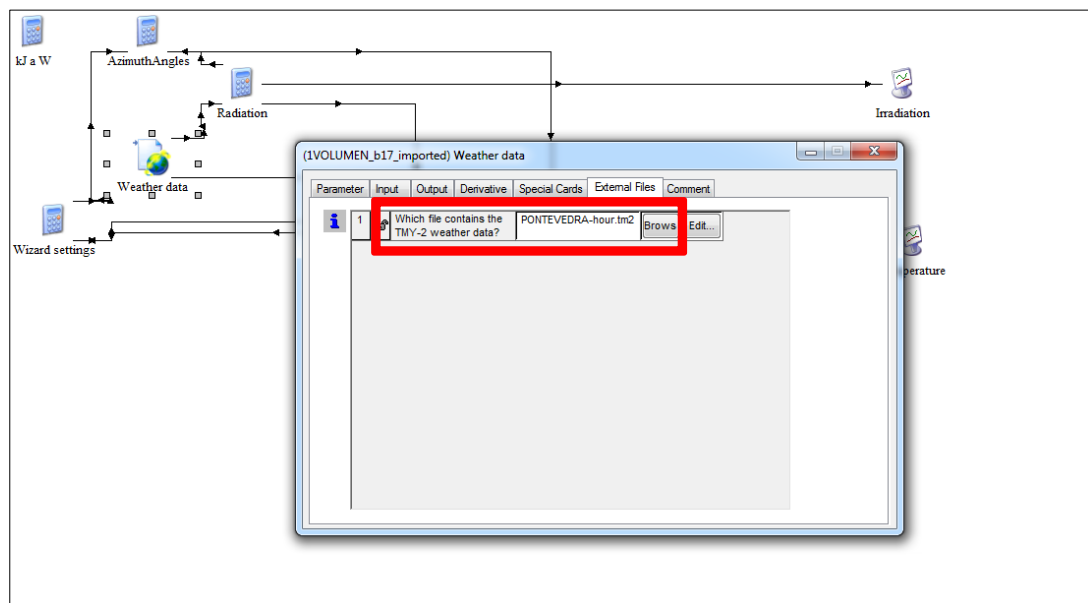


Figura 4-89: Datos meteorológicos de la simulación

Para exportar los datos en documentos .txt, se debe crear el archivo de texto en la carpeta de trabajo y, una vez creado, introducirlo como archivo externo (*External file*) en el type deseado.

Una vez finalizada la configuración de la simulación, se debe configurar el volumen creado para la simulación, para ello se utiliza la aplicación *TRNBuild*. Se asignó material a cada una de las paredes (Figura 4-90). Se utilizó el mismo material y grosor que el casco del patrullero Tabarca siendo Acero A de 7 centímetros (Figura 4-91).

En las simulaciones se empleará un cerramiento de Acero A de 7 centímetros, empleando de esta manera un cerramiento que aisle más del exterior pudiendo así comprobar la influencia de las variables meteorológicas con cerramientos de mayor tamaño a los reales.

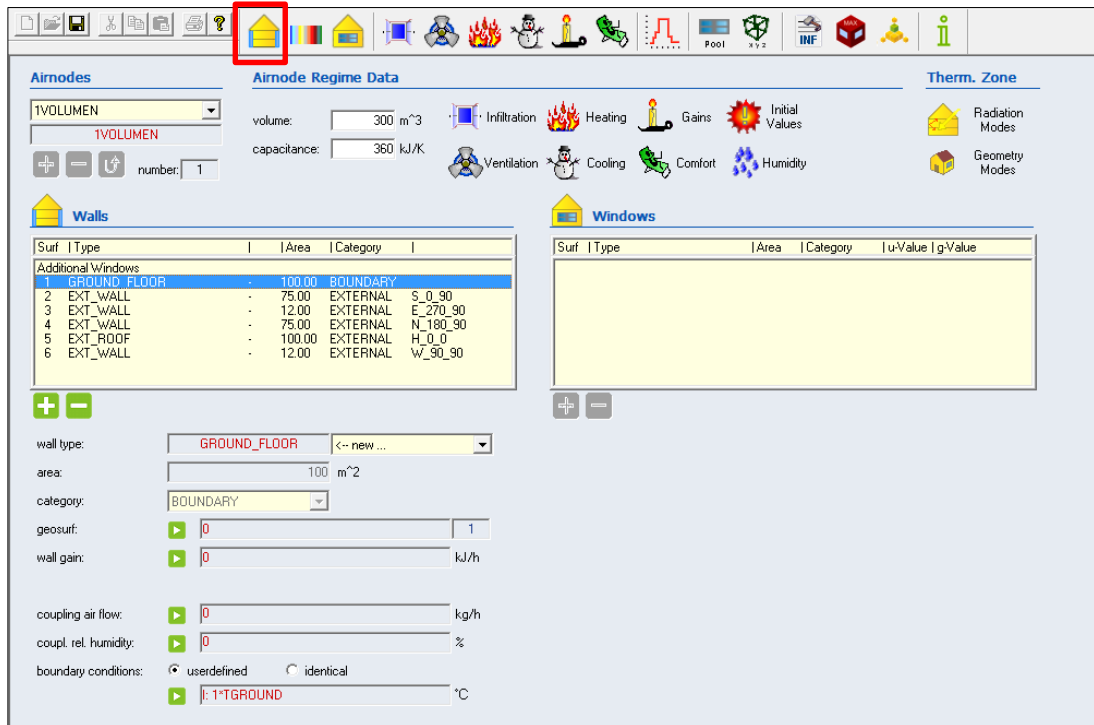


Figura 4-90: TRNBuild

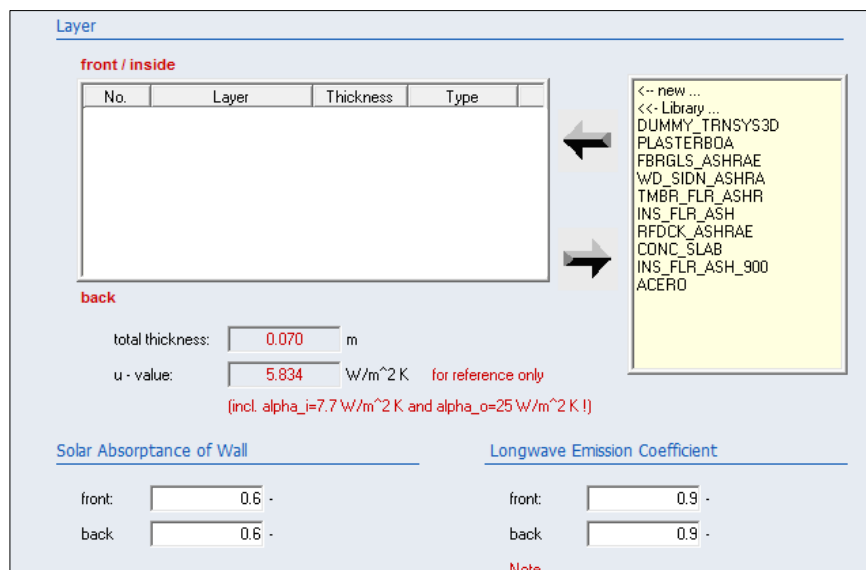


Figura 4-91: Selección de material en TRNBuild

En cuanto a la selección de material, se deben estudiar las propiedades térmicas como densidad, calor específico y conductividad térmica. A continuación se definen estas propiedades [29]:

- Densidad (ρ): masa de material por unidad de volumen (*density* en la Figura 4-92).
- Calor específico (c): cantidad de energía necesaria para aumentar en 1 ° C la temperatura de 1 kg de materia. Indica la mayor o menor dificultad que presenta una sustancia para experimentar cambios de temperatura bajo el suministro de calor. Los materiales que presenten un elevado calor específico serán buenos aislantes. Sus unidades del Sistema Internacional son J/ (kg · K), aunque también se

suele presentar como kcal/ (kg · ° C); siendo 1 cal = 4,184 J (*capacity* en la Figura 4-92).

- Conductividad térmica (λ): capacidad de un material para transferir calor. La conducción térmica es el fenómeno por el cual el calor se transporta de regiones de alta temperatura a regiones de baja temperatura dentro de un mismo material o entre diferentes cuerpos. Las unidades de conductividad térmica en el Sistema Internacional son W/ (m·K) (*conductivity* en la Figura 4-92).

Layer	
new layer type:	ACERO
thickness:	m
group:	Metales
material:	Acero
Massive Layer	
conductivity:	180.000000 kJ / h m K
capacity:	0.500000 kJ / kg K
density:	7800.000000 kg / m ³

Figura 4-92: Propiedades Acero de la biblioteca *TRNBuild*

Los Aceros Navales poseen un valor de Conductividad Térmica entre 40 y 60 W/ (m·K) [42]. En las bibliotecas de *TRNBuild*, al no disponer de Acero Naval se emplea Acero ya que las propiedades térmicas (Tabla 4-2) son similares a las del Acero Naval.

Tabla 4-2: Propiedades térmicas del Acero

Densidad (ρ)	$\rho = \frac{m}{V} \text{ kg/m}^3$	$\rho = 7800 \text{ kg/m}^3$
Calor específico (c)	$c = \frac{Q}{m} \text{ J/kgK}$	$c = 500 \text{ J/kgK}$
Conductividad térmica (λ)	$\lambda = \frac{\dot{q}}{ \nabla T } \text{ W/(K} \cdot \text{m)}$ $\dot{q}$, es el flujo de calor ∇T, es el gradiente de temperatura	$\lambda = \frac{180 \frac{\text{kJ}}{\text{h(mK)}} \cdot 1000}{3600} = 50 \frac{\text{W}}{\text{mK}}$ 1 W = 1 J/s

Una vez seleccionado el material, se climatiza la zona. *TRNBuild* realiza balances energéticos y calcula la cantidad de energía (potencia térmica) necesaria para que la zona no pueda descender o aumentar nunca por debajo y encima de las temperaturas de consigna. La potencia calorífica sirve para limitar la potencia del sistema ideal.

Para ello se fijaron dos temperaturas de consigna: 20°C para calefacción y 26°C para refrigeración, las cuales vienen seleccionadas por defecto en el programa, cumpliendo con la normativa española UNE-EN ISO 13790 para la eficiencia energética de los edificios [43]. Se varió la temperatura inferior de consigna respecto a las que vienen por defecto en el programa de simulación, empleando 15 ° C en lugar de 20 ° C, con la finalidad de obtener un mayor margen de temperaturas en el interior y así poder analizar de manera más clara la evolución de temperaturas hasta alcanzar la temperatura máxima de consigna.

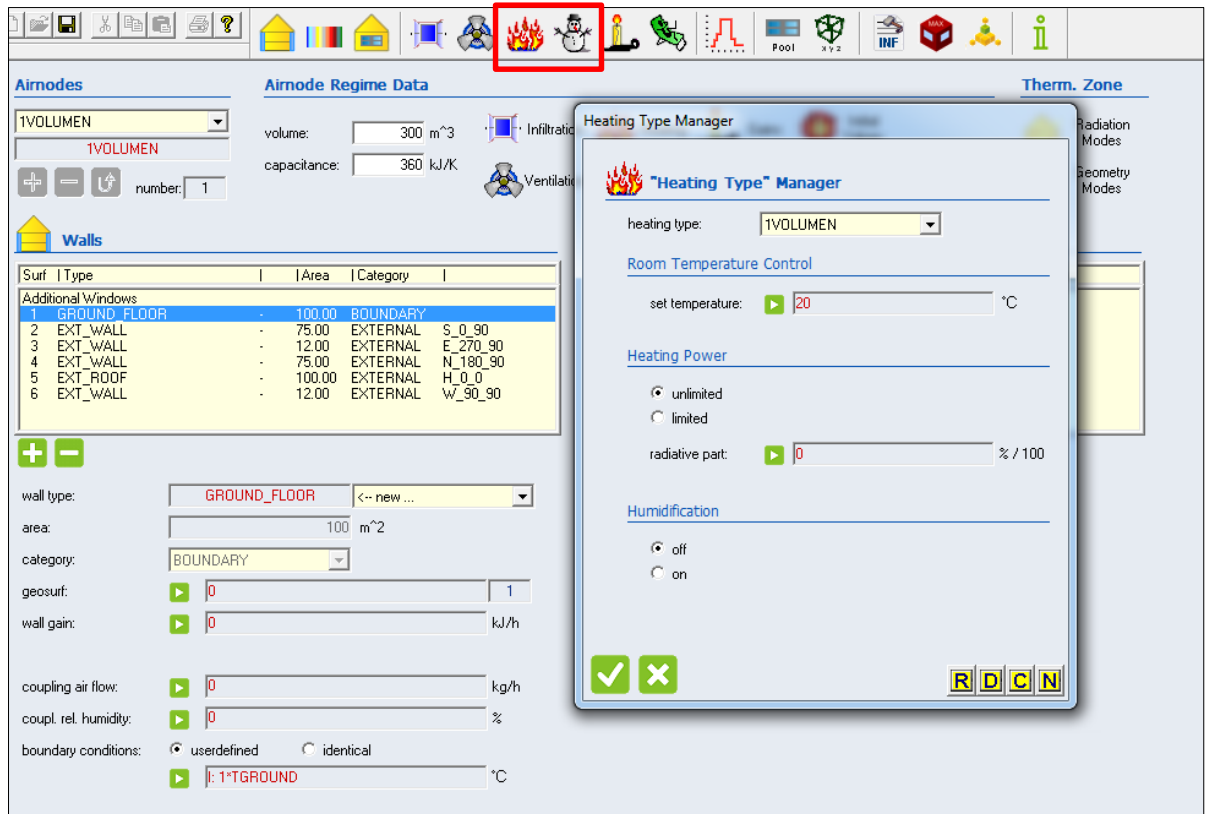


Figura 4-93: Climatización de la zona con temperaturas de consigna

Una vez creadas las condiciones de climatización deben aplicarse al volumen que se emplea en la simulación (Figura 4-93 y Figura 4-94) activándolas en cada una de las diferentes zonas con las opciones *Heating* y *Cooling*. Una vez finalizado esto, se ejecuta la simulación en *Simulation Studio*.

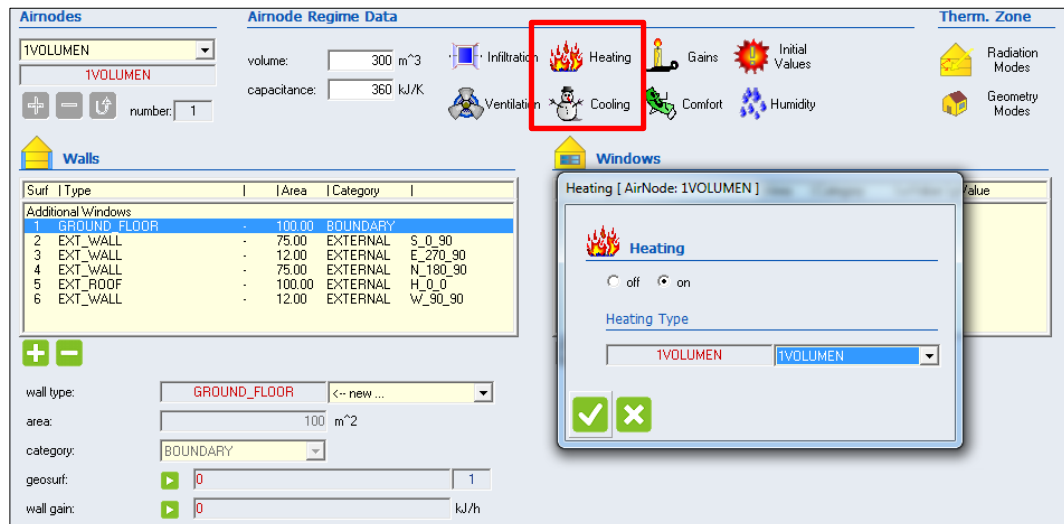


Figura 4-94: Aplicación de la climatización

4.6.1.2.2 Resultados simulación con una zona térmica

Una vez ejecutada la simulación, se analizan los resultados obtenidos (graficas y archivos de texto) que se han creado con los types *System_Printer*.

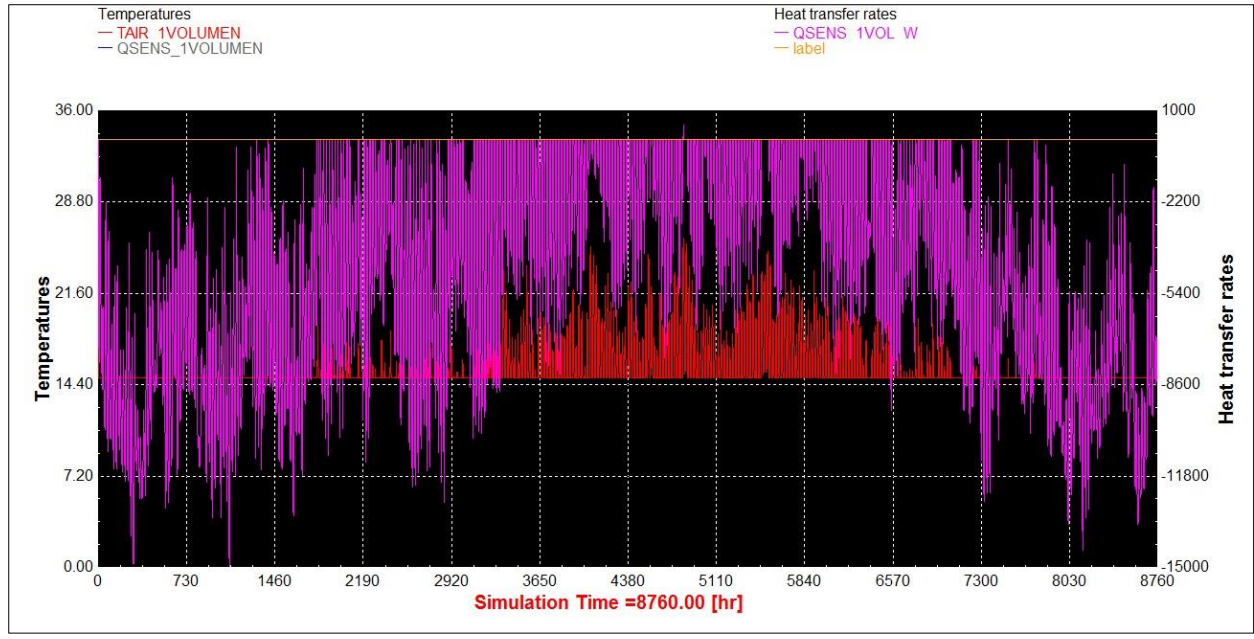


Figura 4-95: Resultados simulación de un volumen

La Figura 4-95 representa los datos que figuran como Inputs en el type de Temperaturas y muestra las temperaturas y los aportes de calor de un año completo (8760 horas).

Calor sensible (Ecuación 4-3) es aquel que recibe un cuerpo o un objeto y hace que aumente su temperatura sin afectar su estructura molecular y por lo tanto su estado [20]. En general, se ha observado experimentalmente que la cantidad de calor necesaria para calentar o enfriar un cuerpo es directamente proporcional a la masa del cuerpo y a la diferencia de temperaturas. La constante de proporcionalidad recibe el nombre de calor específico. Si el proceso se efectúa a volumen constante [44]:

Ecuación 4-3: Calor sensible

$$Q = \Delta U = nC_v\Delta T$$

En donde U representa la energía interna del sistema, n son las moles de la sustancia y C_v es el calor específico a volumen constante. Los valores de calor específico varían también con la temperatura ambiente y el estado físico de agregación de las sustancias.

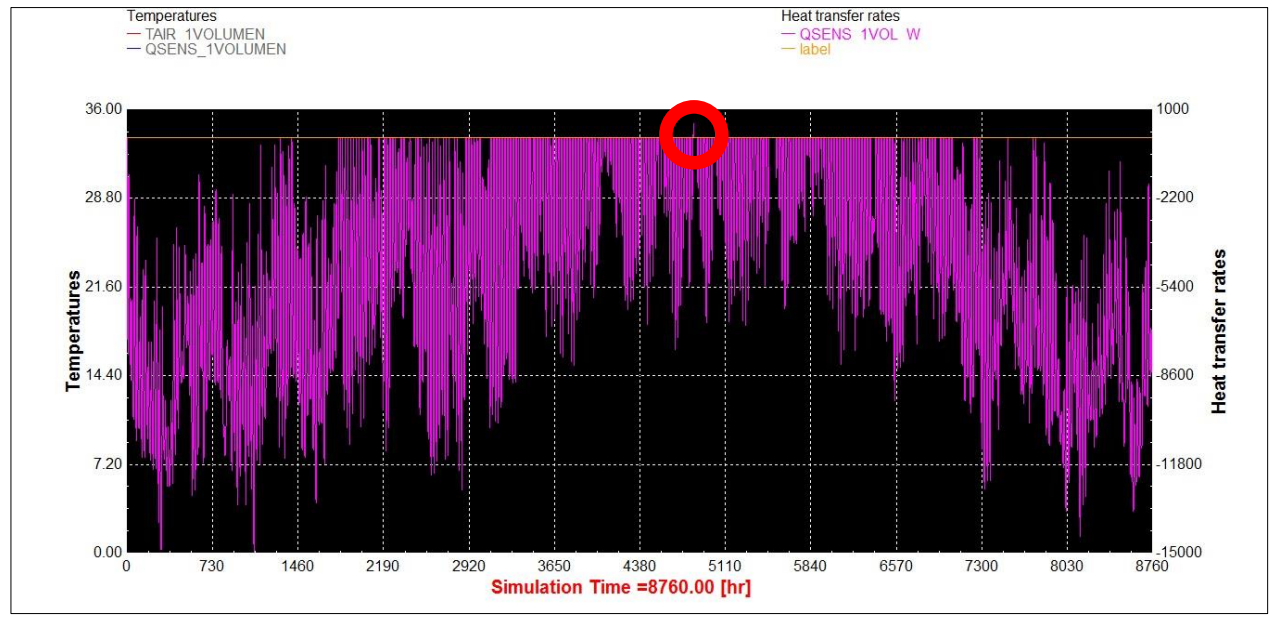


Figura 4-96: Resultados de intercambio de calor de la simulación de un volumen

Analizando los datos de energía de la construcción, se puede observar que se necesita un aporte energético en la edificación para mantener las temperaturas que se han seleccionado como temperaturas de consigna (15°C y 26°C) durante todo el año, con las pequeñas excepciones de determinados momentos durante las horas centrales de la simulación (señalado con un círculo rojo en la Figura 4-96). Estas horas corresponden a los meses de verano (junio, julio y agosto), en las que la construcción cede calor al medio por alcanzarse la temperaturas de consigna máxima establecida (26°C en este caso), tratándose esa hora concreta del 19 de julio a las 14:00 pm, día y hora con una temperatura ambiente muy elevada como se muestra en la Figura 4-97.

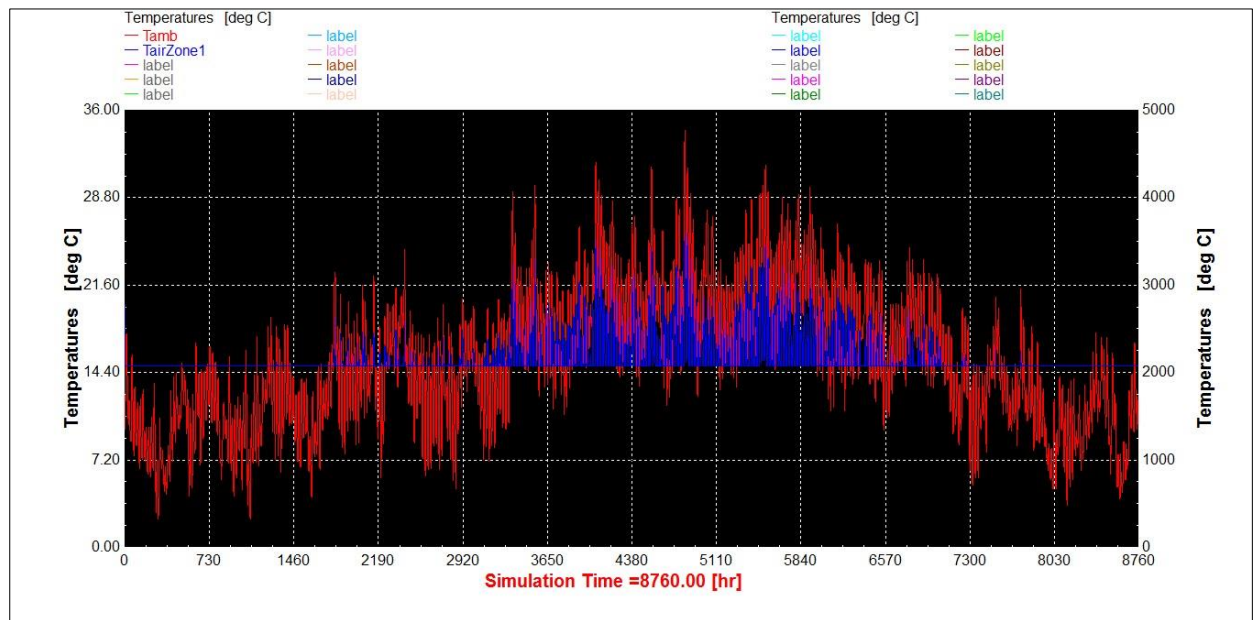


Figura 4-97: Resultados de temperaturas de la simulación de un volumen

Con los datos obtenidos en la Figura 4-97 se observa las diferencias de temperaturas entre las temperaturas del interior de la construcción (color azul) (siendo TairZone1 la temperatura del aire en el interior de la zona térmica) y la temperatura del aire del ambiente (color rojo) (Tair) obtenida esta última de la base de datos de *Meteonorm* en la ubicación estudiada. Se observa que la tempera

ambiente durante los meses iniciales y finales del año es inferior a la temperatura de 15° C fijada como temperatura mínima de la construcción, por lo que se explica con esto los resultados de necesidades térmicas durante estos meses. En cambio, durante los meses centrales de la simulación (junio, julio y agosto) se puede observar el aumento de las temperaturas del ambiente por lo que la temperatura interior aumenta también superando la mínima establecida y llegando incluso a la temperatura máxima establecida (26 ° C) como se puede observar en la Figura 4-96 por lo que cambiaba de signo el aporte de energía significando esto que la construcción cedía calor al ambiente.

Tabla 4-3: Resumen de resultados de la simulación para un volumen

	Temperatura ambiente (° C)	Temperatura interna de la edificación (° C)	Aporte de energía (W)
Enero	6.12	15.01	-8447.86
Febrero	5.85	15.00	-7529.01
Marzo	7.25	15.14	-5753.05
Abril	8.24	15.12	-4790.95
Mayo	10.80	15.71	-3120.58
Junio	13.90	16.63	-1739.22
Julio	15.02	17.05	-1323.55
Agosto	15.07	16.96	-1292.24
Septiembre	13.78	16.11	-2101.18
Octubre	12.44	15.24	-3680.20
Noviembre	8.31	15.01	-7034.36
Diciembre	6.50	15.00	-8435.18

En la Tabla 4-3 figuran los resultados de las temperaturas promedio de los diferentes meses del año, así como los intercambios de calor producidos. Estos promedios se calcularon con los archivos de texto exportados con los types *System_Printer* en la simulación. Los datos exportados de *Trnsys* se tratan de datos horarios, esto es porque se indicó un *step time* de una hora en las condiciones de la simulación. Se realiza la simulación durante un periodo de tiempo establecido, siendo en este caso, un año completo (8760 horas). Los datos promedios de cada mes se calcularon a partir de los datos horarios, con ayuda del programa de cálculo Excel. Se calcula la temperatura ambiente en la primera columna y la temperatura interna de la construcción en la segunda columna. Como último dato de estudio se añade el calor sensible medido en vatios (W) (*QSENS_1VOL_W*). Los datos proporcionados por *Trnsys* de intercambio de calor, se obtienen por defecto en kilojulios por hora (kJ) por lo que se añade un type calculadora con el que se realiza el cambio de unidades.

Al realizarse la simulación para un único volumen, no se produce intercambio de calor entre zonas térmicas, por lo que los datos de *QSENS* obtenidos de *TRNSYS*, equivalen a la energía que se debe aportar a la edificación para mantener las temperaturas de consigna, entre 15°C y 26°C. Siguiendo el manual de *TRNSYS*, *QSENS* se puede definir como la demanda de energía sensible en una zona térmica, siendo su signo negativo cuando se trate de necesidades térmicas de calor y positivo en caso de que se trate de demandas térmicas de frío [20].

Se observa claramente en la parte coloreada de azul de la Tabla 4-1, que las temperaturas en los meses de más frío (noviembre, diciembre, enero y febrero) en los que la temperatura de la edificación está en el límite de lo programado, se necesita aportar gran cantidad de energía por hora para mantener las condiciones de temperatura definidas en la zona térmica, llegando, como media horaria, a valores de 8447,86 W para el mes de enero.

Contrariamente, en los meses de más calor (junio, julio y agosto), la energía necesaria para climatizar el volumen es mínima, llegando a ser de signo positivo, es decir, demanda térmica de frío (señalado en Figura 4-96), ya que la temperatura ambiente es muy superior a las temperaturas de los meses de los meses más fríos.

Se puede concluir que el material y los aislantes de la envolvente son importantes en los resultados de la simulación. Al tratarse de una cubierta de acero sin aislantes, la radiación aporta energía a la construcción de manera considerable debido a su orientación respecto a la trayectoria solar (Figura 4-86), llegando a superar apenas sin aporte de energía, las temperaturas establecidas como máximas en el interior de la construcción (26 °C).

4.6.1.2.3 Simulación con dos zonas térmicas

Se realiza una modificación de la Simulación con una zona térmica para analizar la diferencia entre realizar una o más zonas térmicas que tendrán las mismas características y ubicación (Figura 4-87).

Se comienza por dividir el volumen inicial que simulaba la obra viva del patrullero Tabarca, en dos volúmenes iguales en *SketchUp* (Figura 4-98), manteniendo las mismas medidas y volumen de 300 m³ en conjunto.

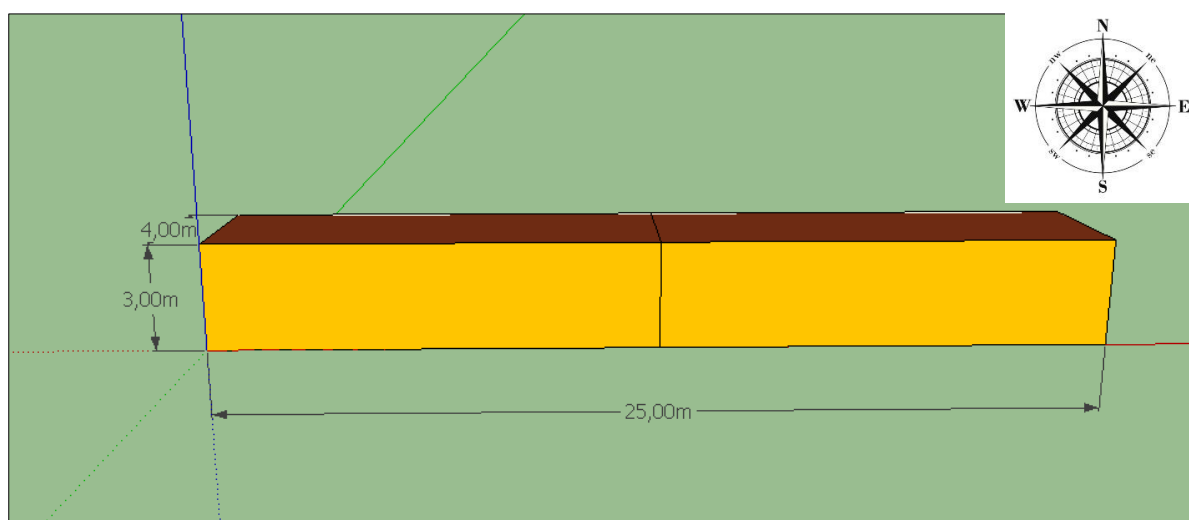


Figura 4-98: Modelo en SketchUp para dos zonas térmicas de 300 m³

Se crea la simulación en *Trnsys*, introduciendo los mismos parámetros característicos de la simulación que en la simulación para un único volumen, siendo estos los datos meteorológicos obtenidos con *Meteonorm* de Pontevedra, aplicando mismos materiales a ambos volúmenes (Figura 4-90 y Figura 4-91) y las mismas condiciones de climatización (Figura 4-93 y Figura 4-94).

Se aplicará un tipo diferente de pared en la unión de los dos volúmenes, siendo este *ADJ_WALL* como pared adyacente, aunque ambas serán del mismo material. Esto se hace para que el programa interprete que dichas paredes no se encuentran en contacto con el exterior, sino que están en contacto entre ellas.

4.6.1.2.4 Resultados simulación con dos zonas térmicas

Para realizar el análisis de los datos obtenidos se emplean igualmente los datos de las gráficas proporcionadas por *Trnsys* y los archivos de texto exportados de los types (Figura 4-99).

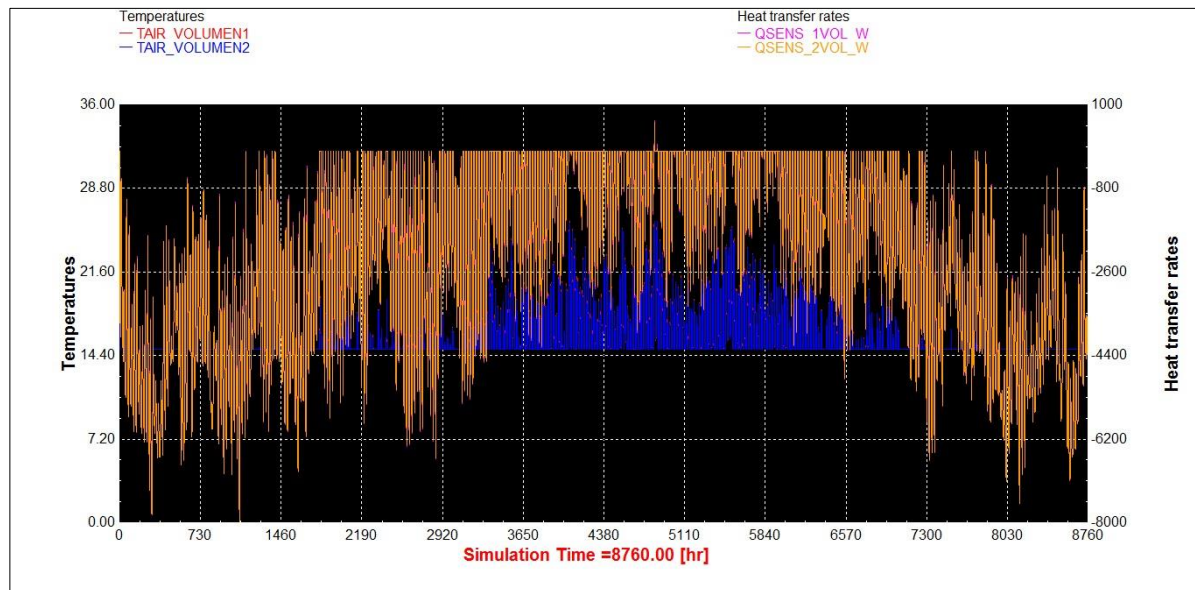


Figura 4-99: Resultados simulación de dos volúmenes

Se obtiene una gráfica similar a la obtenida en la simulación para un único volumen, con la diferencia que en esta simulación se trata de dos volúmenes, se analizan las temperaturas interiores (Eje izquierdo en color azul y rojo) y el intercambio de energía para cada uno de los volúmenes (Eje derecho en color naranja y rosa).

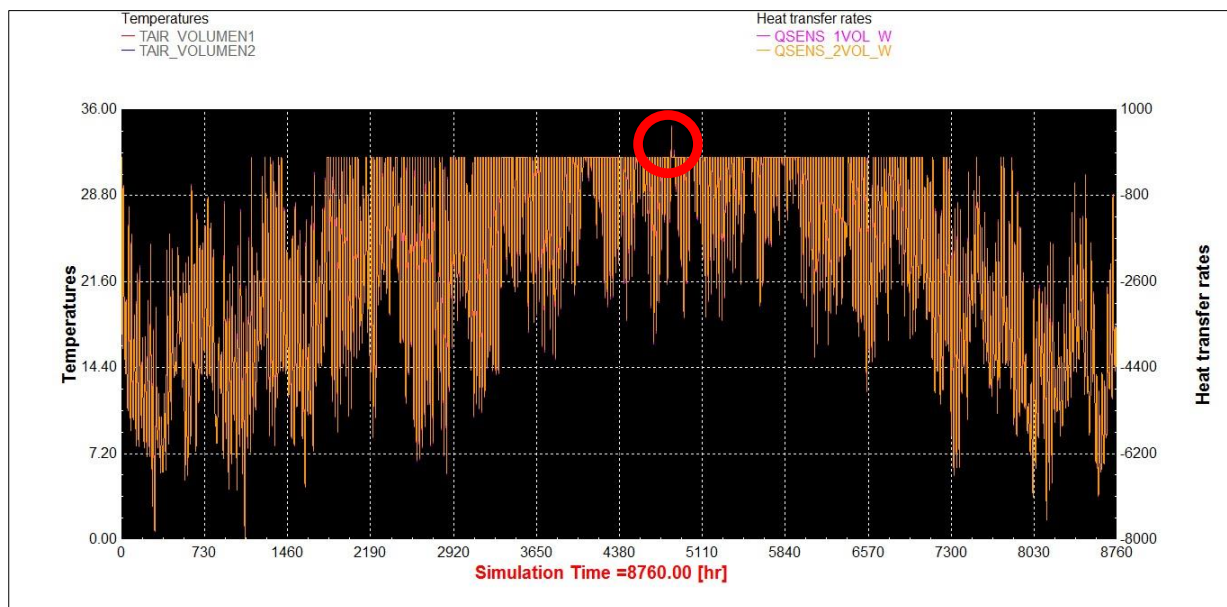


Figura 4-100: Resultados de intercambio de energía de la simulación de dos volúmenes

Se obtienen datos de intercambio de energía (Figura 4-100) y de temperaturas internas de los volúmenes (Figura 4-101), a primera vista, muy similares con la simulación anterior, y muy próximos entre los dos volúmenes a estudiar, por lo que se opta por analizar los datos numéricos exportados.

En el caso de las temperaturas, no se configura la temperatura del aire como output, ya que al ser la misma que en la simulación anterior carece de interés para los resultados. Se observa la diferencia

entre las temperaturas internas de ambos volúmenes ya que es uno de los datos de interés en esta simulación.

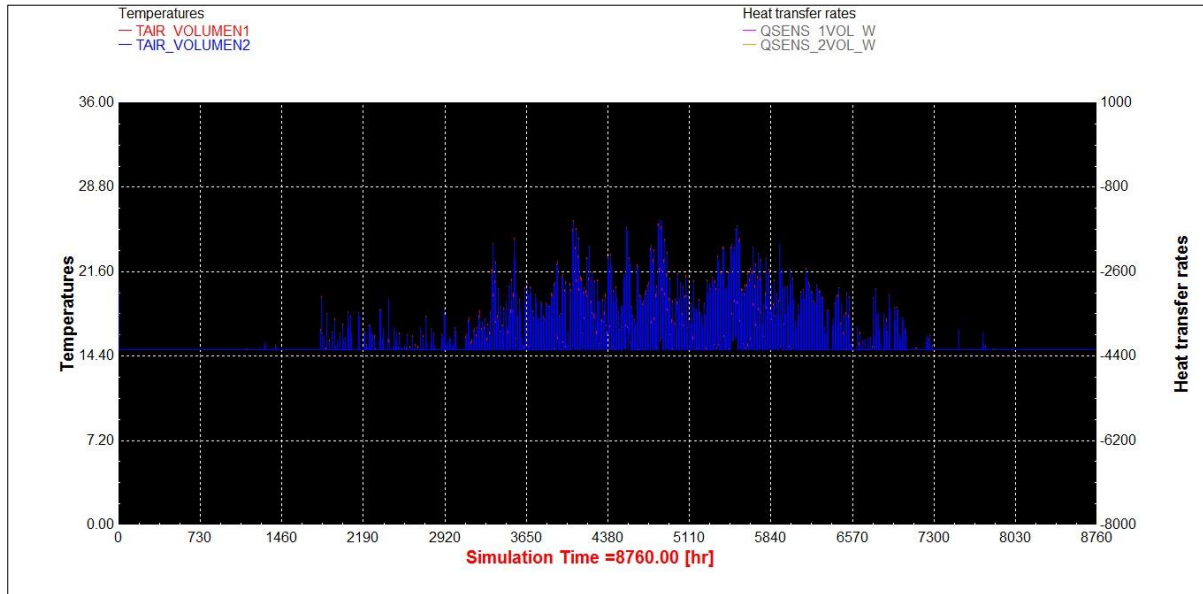


Figura 4-101: Resultados de temperaturas de la simulación de dos volúmenes

En la Tabla 4-4 se comparan y analizan las diferencias entre las temperaturas internas de los volúmenes y los aportes de calor en la edificación. El software exporta los de las temperaturas internas de las diferentes zonas térmicas, y los intercambios de energía de cada uno de los volúmenes.

Una vez se computan los datos promedio por meses, se calcula con ayuda de una hoja de Excel, el porcentaje de variación que existe entre los datos de ambos volúmenes, siendo cada uno de ellos una zona térmica.

Tabla 4-4: Resumen de resultados de la simulación para dos volúmenes

	Temperatura interna 1	Temperatura interna 2	% variación de temperatura	Aporte de calor 1	Aporte de calor 2	% variación de aporte de calor
Enero	15.01	15.01	0.000	-4364.28	-4361.49	0.064
Febrero	15.00	15.00	0.006	-3852.80	-3852.41	0.010
Marzo	15.19	15.18	0.046	-2894.01	-2890.63	0.117
Abril	15.18	15.17	0.034	-2361.35	-2352.20	0.389
Mayo	15.88	15.87	0.100	-1497.55	-1487.99	0.643
Junio	16.94	16.91	0.175	-801.30	-787.55	1.746
Julio	17.39	17.37	0.111	-589.09	-577.52	2.004
Agosto	17.29	17.27	0.108	-580.58	-573.67	1.205

Septiembre	16.32	16.31	0.091	-989.53	-979.94	0.979
Octubre	15.31	15.30	0.072	-1806.43	-1802.95	0.193
Noviembre	15.01	15.01	0.006	-3604.19	-3594.27	0.276
Diciembre	15.00	15.00	0.000	-4361.80	-4361.05	0.017

Para obtener los % de diferencia entre temperaturas y aportes de energía, se aplican las siguientes formulas:

Ecuación 4-4: Porcentaje de diferencia en datos

$$\%Diferencia\ DATO = \frac{Dato_variable_{volumen1} - Dato_variable_{volumen2}}{Dato_variable_{volumen2}} * 100$$

Se observa en la Tabla 4-4 que las diferencias de temperatura entre los dos volúmenes, no supera el 0.2% (señalado en negrita), por lo que se puede considerar despreciable esta diferencia. En cuanto a las diferencias de aporte de energía, se considerarán despreciables las diferencias debido a que no superan el 2.004% en ningún caso (señalado en negrita).

Como conclusión y análisis de datos entre la simulación de uno o varias zonas térmicas, se comparan los datos de temperaturas internas de una zona térmica y de varias, observando la similitud de los datos (Tabla 4-5). En algún caso la pequeña variación de la temperatura interna de una zona térmica (Temperatura 1 Z.T.) con respecto a las dos (Temperatura 1 Z.T. y Temperatura 2 Z.T.), se debe principalmente al tamaño de la construcción a climatizar, siendo inversamente proporcional la temperatura alcanzada con el tamaño de la construcción, siempre y cuando se mantenga la necesidad térmica de la zona.

Tabla 4-5: Comparación de temperaturas internas (° C)

Meses del año	Temperatura 1 Z.T.	Temperatura 1 2 Z.T.	Temperatura 2 2 Z.T.
Enero	15.01	15.01	15.01
Febrero	15.00	15.00	15.00
Marzo	15.14	15.19	15.18
Abril	15.12	15.18	15.17
Mayo	15.71	15.88	15.87
Junio	16.63	16.94	16.91
Julio	17.05	17.39	17.37
Agosto	16.96	17.29	17.27
Septiembre	16.11	16.32	16.31
Octubre	15.24	15.31	15.30
Noviembre	15.01	15.01	15.01

Diciembre	15.00	15.00	15.00
-----------	-------	-------	-------

En cuanto a las necesidades térmicas de las zonas (Tabla 4-6), se realiza la comparación entre el aporte energético con un único volumen, y la suma de los aportes energéticos considerando dos volúmenes. Se obtiene como resultado una proximidad en resultados, lo cuál indica que el aporte de energía necesario no se ve afectado por el número de zonas térmicas que se realicen si se mantiene la misma orientación respecto al sol, la misma ubicación de la construcción y el mismo tamaño y tipo de cerramientos.

Tabla 4-6: Comparación de necesidades térmicas (W)

Meses del año	Aporte de energía para 1 Z.T.	Aporte de energía 1 para 2 Z.T.	Aporte de calor 2 para 2 Z.T.	Aporte de energía total 2 Z.T.
Enero	-8447.86	-4364.28	-4361.49	-8725.77
Febrero	-7529.01	-3852.80	-3852.41	-7705.21
Marzo	-5753.05	-2894.01	-2890.63	-5784.64
Abril	-4790.95	-2361.35	-2352.20	-4713.55
Mayo	-3120.58	-1497.55	-1487.99	-2985.54
Junio	-1739.22	-801.30	-787.55	-1588.85
Julio	-1323.55	-589.09	-577.52	-1166.61
Agosto	-1292.24	-580.58	-573.67	-1154.25
Septiembre	-2101.18	-989.53	-979.94	-1969.47
Octubre	-3680.20	-1806.43	-1802.95	-3609.38
Noviembre	-7034.36	-3604.19	-3594.27	-7198.46
Diciembre	-8435.18	-4361.80	-4361.05	-8722.85

4.6.1.2.5 Conclusiones

Como conclusión de Utilizar una o varias zonas térmicas para un mismo volumen que se obtienen resultados muy similares de temperaturas internas y necesidades energéticas cuando compartan ubicación, orientación y cerramientos, por lo que basandonos en dicha demostración se empleará una zona térmica para representar la obra viva del patrullero Tabarca.

4.6.1.3 Temperatura del agua del mar

Se realizan simulaciones térmicas para comprobar la influencia que tiene la temperatura del mar en la temperatura interna de los buques. Para ello, se consideran dos casos, siendo el primero de ellos un supuesto verano donde las temperaturas del agua son elevadas, y su opuesto donde las temperaturas para un supuesto invierno son inferiores.

Se comienza por crear un volumen que simulará la obra viva del patrullero Tabarca, como en otras simulaciones anteriores se emplea un volumen de 300 m^3 . En este caso se orienta de la misma manera que el caso real, siendo esta orientación un Norte-Sur (línea de color verde de *SketchUp* y parte superior de la imagen satélite de la Figura 4-102 indican el Norte).

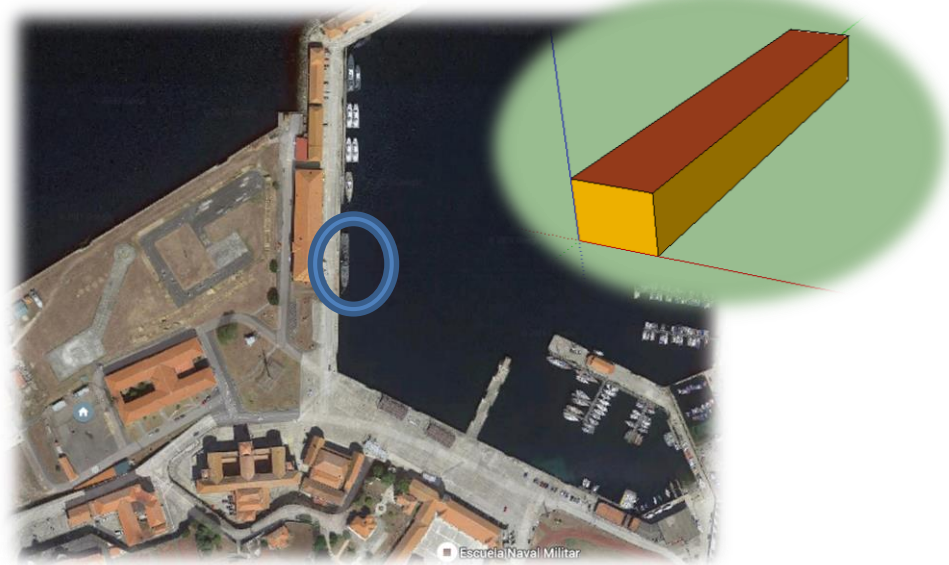


Figura 4-102: Volumen de *SketchUp* simulando al patrullero Tabarca [8]

En la creación del archivo de *SketchUp*, se deben configurar las paredes de esta construcción para poder incluir el archivo de temperaturas del mar, y de esta manera simular que las paredes en lugar de estar en contacto con el aire ambiente, lo estarán con un fluido diferente. Se pondrán las paredes como *BND_WALL* y en la opción de *Outside Boundary Condition* se seleccionará la opción *Other Side Coefficients* y *BOUNDARY=INPUT 1*TBOUNDARY*. La única cara que no estará sumergida en agua, será la parte superior y se pondrá *EXT_ROOF*, por tanto a esta cara le afectarán las condiciones ambientales importadas de la base de datos de *Meteonorm*.

Se crea una simulación en *Trnsys* (Figura 4-103), de la misma manera que el apartado 4.6.1.2 Utilizar una o varias zonas térmicas para un mismo volumen. Se debe introducir el archivo meteorológico exportado de *Meteonorm* en el type *Weather Data* (PONTEVEDRA-hour.tm2), editar el type *Building* asignando material y grosor a las paredes (Acero 7 cm) y configurando la calefacción (20°C) y la refrigeración (26°C) de la zona térmica.

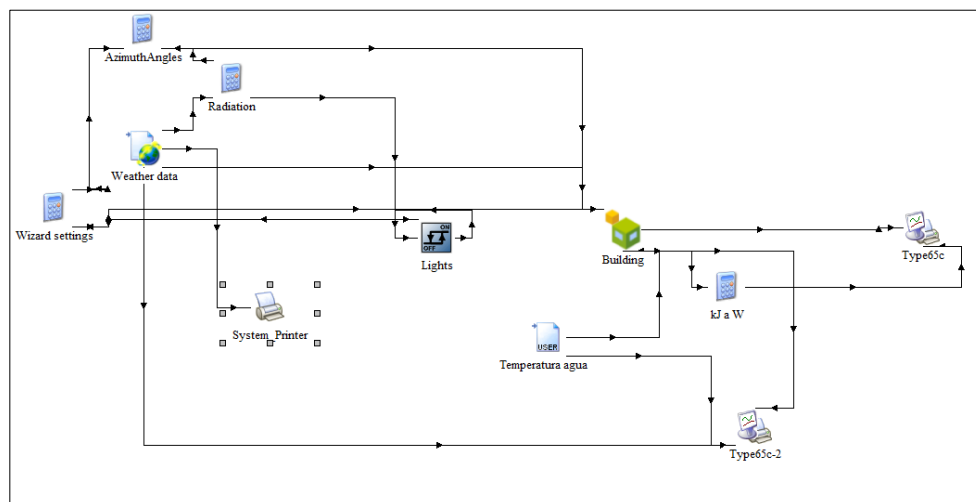


Figura 4-103: Simulación Temperatura del mar en *Trnsys*

Se añade un type (type 9b) para introducir en la simulación los datos de temperatura del mar a partir de un archivo .txt. Se debe configurar el tiempo para efectuar la lectura de cada dato, se pone 30.416 para efectuar una lectura cada mes del año (Figura 4-104), es decir, los datos de temperaturas del agua de mar que se incluyen en la simulación son datos medios mensuales. Éste dato se modifica en la pestaña *Parameter*, en el punto 4 (*Time Interval of Data*). Para que el type pueda efectuar la lectura de los datos, se debe introducir como *External File* el documento de texto creado con las supuestas temperaturas medias mensuales del mar. Para crear este archivo es necesario introducir 13 datos, siendo el primero igual al último para que el type pueda realizar una lectura repetitiva del archivo en caso de querer una simulación de más de un año.

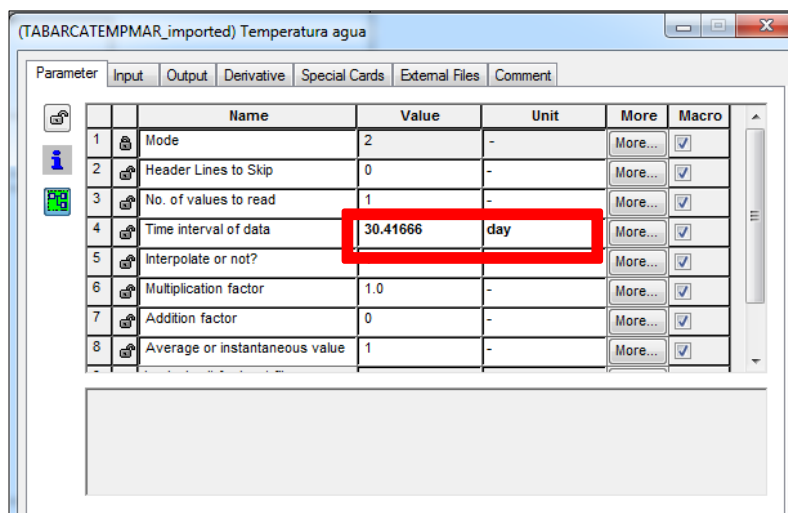


Figura 4-104: Configuración type 9b

Para introducir el dato de temperatura del mar en la simulación debemos establecer las uniones entre outputs e inputs de los diferentes types, uniendo en este caso, el *Output 1* (temperatura del archivo de temperaturas creado como se muestra en la Figura 4-105) con *TBOUNDARY* del volumen. *TBOUNDARY* se emplea para indicar que la pared está sumergida en agua con la temperatura indicada por nosotros en lugar de en condiciones ambientales.

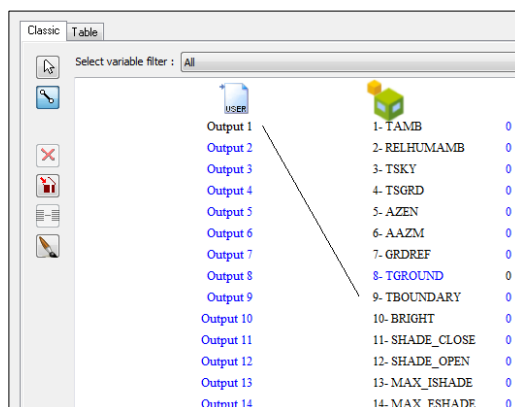


Figura 4-105: Introducir dato de temperatura del mar en la simulación

Después de crear la simulación se deben realizar los dos supuestos planteados (Temperaturas del mar elevadas en verano y temperaturas del agua bajas en invierno). Para ellos todos los pasos anteriores son comunes. Se introduce como *External File* en el type 9b el archivo creado para verano, y se crean archivos de texto para obtener los resultados al exportar datos (type 65c y *System_Printer*). Se debe cambiar el archivo del type 9b y crear archivos para los datos a exportar en el caso de la simulación de invierno.

4.6.1.3.1 Influencia de la temperatura del agua de mar en la simulación

Se representan en *Trnsys* los resultados de temperaturas en gráficas, incluyendo la temperatura ambiente, temperatura del mar y temperatura interna del Tabarca en la simulación. Se puede observar la diferencia de las temperaturas internas del Tabarca en el supuesto invierno y verano (color azul de las Figura 4-106 y Figura 4-107).

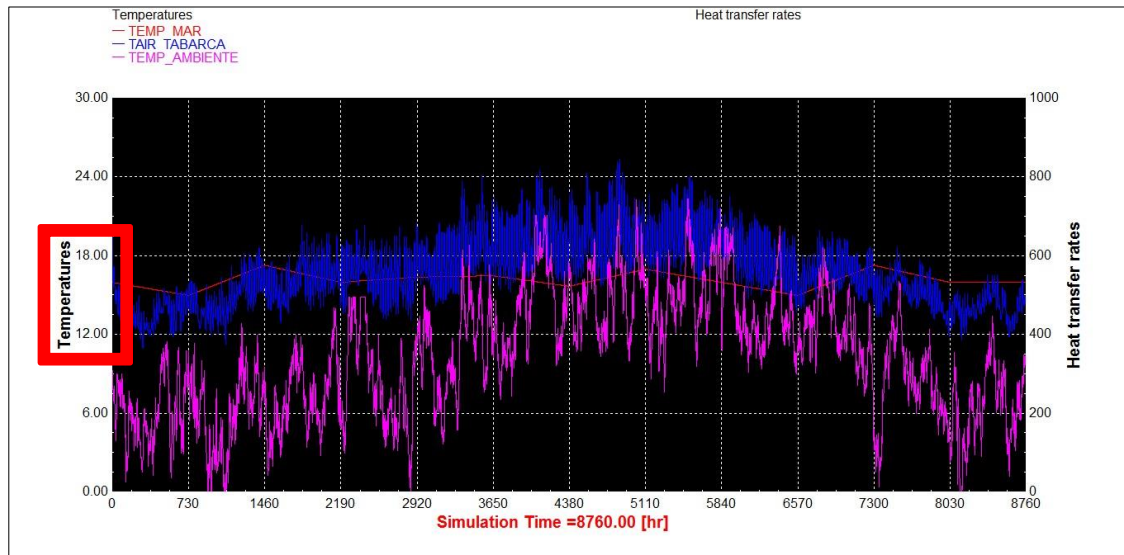


Figura 4-106: Resultados de temperaturas para verano

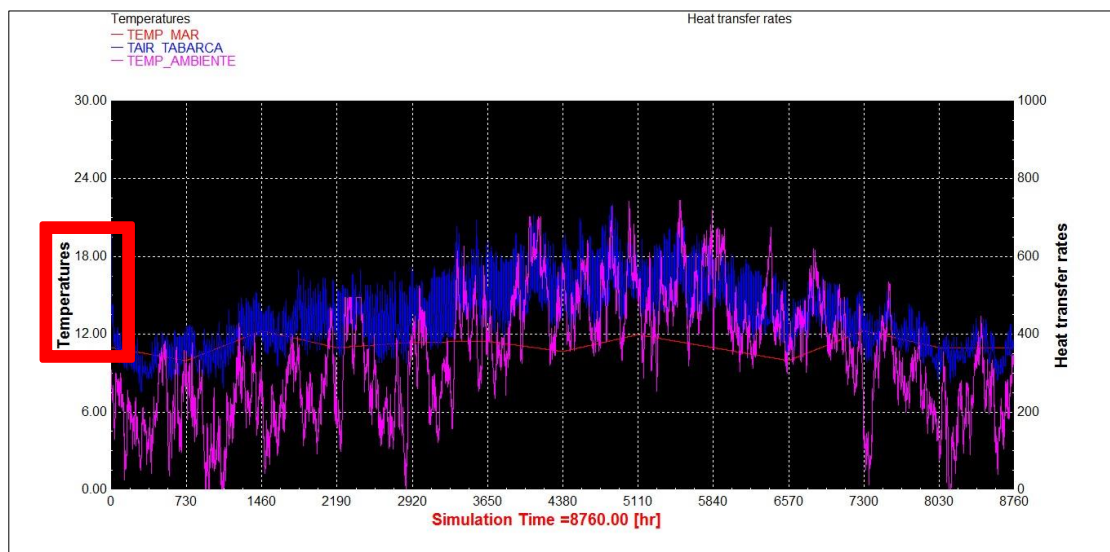


Figura 4-107: Resultados de temperaturas para invierno

Para el análisis de resultados se utilizan tablas creadas en Excel (Tabla 4-7). Se calculan los promedios mensuales de las diferentes temperaturas para realizar la comparación entre verano e invierno y así poder analizar la influencia de la temperatura del mar en las simulaciones.

Tabla 4-7: Análisis de la influencia de la temperatura del mar

	T.ambiente	T.mar_frio	T.mar_calor	T.interna_frio	T.interna_calor
Enero	6.12	10.49	15.49	10.43	13.76
Febrero	5.85	11.21	16.21	11.42	14.77

Marzo	7.25	11.65	16.65	12.84	16.19
Abril	8.24	11.19	16.19	13.21	16.56
Mayo	10.80	11.45	16.45	14.72	18.06
Junio	13.90	11.12	16.12	15.79	19.14
Julio	15.02	11.32	16.32	16.33	19.67
Agosto	15.07	11.52	16.52	16.24	19.59
Septiembre	13.78	10.50	15.50	14.49	17.83
Octubre	12.44	11.15	16.15	13.43	16.78
Noviembre	8.31	11.65	16.65	11.85	15.21
Diciembre	6.50	11.00	16.00	10.73	14.09

Se incluyen en la Tabla 4-7 la temperatura ambiente, que afectará a la cara superior del Tabarca, las temperaturas del mar en los dos supuestos, creados para la simulación poniendo valores aleatorios con una diferencia de 5 ° C al mes entre las temperaturas de verano e invierno (en la simulación se les llamaron calor y frío), y las temperaturas internas del Tabarca que serán los resultados de la simulación.

Para realizar el análisis de resultados, debemos calcular el porcentaje de variación entre los datos de temperatura del mar y también entre las temperaturas internas obtenidas. Para ello empleamos la Ecuación 4-5:

Ecuación 4-5: Porcentaje de diferencia en datos

$$\%Diferencia\ DATO = \frac{Dato_variable_{volumen1} - Dato_variable_{volumen2}}{Dato_variable_{volumen2}} * 100$$

Se obtienen los datos de la Tabla 4-8 como objeto de análisis.

Tabla 4-8: Resultados simulación con temperaturas del mar

	% T. mar	% T. interna
Enero	32.274	24.164
Febrero	30.844	22.706
Marzo	30.024	20.696
Abril	30.886	20.232
Mayo	30.401	18.532
Junio	31.015	17.478
Julio	30.640	17.000

Agosto	30.265	17.076
Septiembre	32.252	18.768
Octubre	30.960	19.965
Noviembre	30.036	22.050
Diciembre	31.250	23.809
PROMEDIO	30.904	20.206

4.6.1.3.2 Conclusiones

Se puede observar en la Tabla 4-8 que la variación de 5 ° C de la temperatura del mar entre los dos supuestos (T.mar_frio y T.mar_calor), produce aproximadamente un 30.9% de variación. Con esta diferencia en las temperaturas se obtiene un 20.2% de variación entre las temperaturas internas. Con estos porcentajes se puede observar que la temperatura de la obra viva del buque está afectada por la temperatura del mar de manera considerable, por lo que debe considerarse factor importante y estudiarlo. Para ello mediremos la temperatura en la dársena de la Escuela Naval Militar, donde permanece atracado el patrullero Tabarca en sus estancias en puerto, para así poder realizar de manera más precisa las simulaciones térmicas.

4.6.2 Medición de temperatura interna del buque

4.6.2.1 Sensores utilizados

Para obtener las diferentes temperaturas, distribuimos sensores a lo largo de la obra viva del Tabarca, para obtener datos lo más exacto posibles sobre las temperaturas de confort y climatización de dicha zona [45].

4.6.2.1.1 Sensor de temperatura y humedad ds1923-f5#

El DS1923 (Figura 4-108 y Figura 4-109) es un sensor que se emplea para la medición de temperatura y/o humedad de manera autónoma. Dispone de una sección de memoria protegida, para almacenamiento de mediciones realizadas con la frecuencia de muestreo programada, a intervalos equidistantes desde 1s hasta 273 horas. En el presente trabajo se emplearán los sensores de 4096 de 16 bits. Además de esto, hay 512 bytes de SRAM para almacenar información específica a la aplicación y 64 bytes para los datos de calibración. Si la temperatura o humedad abandonan un rango definido por el usuario, ya sea por debajo o por encima de valores permitidos, el DS1923 registrará ese evento como una alerta. Puede ser programado para iniciar el registro de datos inmediatamente o activarse tras una alarma de temperatura fijada por el usuario. Los datos son transferidos a través del protocolo 1-Wire®, que requiere un solo cable además del contacto de tierra. Su robusta carcasa de acero inoxidable es altamente resistente a los ambientes agresivos, la suciedad, humedad, y golpes. Tiene unas dimensiones estándar F5 para la lectura de datos. Es de fácil montaje sobre cualquier objeto. Tiene una precisión de temperatura de $\pm 0.5^{\circ}\text{C}$ desde -10°C a $+65^{\circ}\text{C}$ con corrección por software.

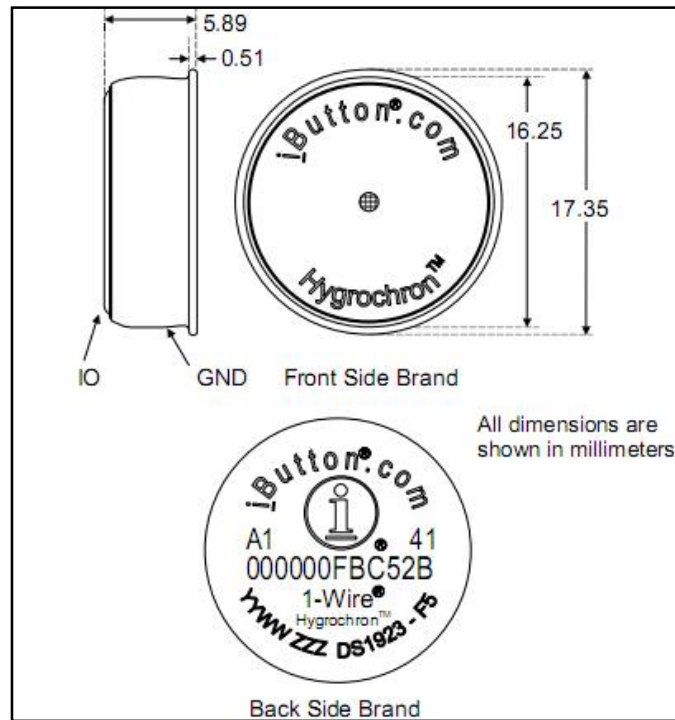


Figura 4-108: Sensor de temperatura y humedad ds1923-f5# [45]



Figura 4-109: Sensor de temperatura y humedad ds1923-f5# [45]

4.6.2.1.2 Sensor de temperatura ds1922l-f5

El DS1922L (Figura 4-110) es un sensor que se emplea para la medición únicamente de temperatura. Dispone de una sección de memoria protegida, para almacenamiento de mediciones realizadas con la frecuencia de muestreo programada, a intervalos equidistantes desde 1s hasta 273 horas. En el presente trabajo se emplearán los sensores de 4096 de 16 bits. Además de esto, hay 512 bytes de SRAM para almacenar información específica a la aplicación y 64 bytes para los datos de calibración. Si la temperatura o humedad abandonan un rango definido por el usuario, ya sea por debajo o por encima de valores permitidos, registrará ese evento como una alerta. Puede ser programado para iniciar el registro de datos inmediatamente o activarse tras una alarma de temperatura

fijada por el usuario. Los datos son transferidos a través del protocolo 1-Wire®, que requiere un solo cable además del contacto de tierra. Su robusta carcasa de acero inoxidable es altamente resistente a los ambientes agresivos, la suciedad, humedad, y golpes. Es de fácil montaje sobre cualquier objeto. Tiene una precisión de temperatura de $\pm 0.5^{\circ}\text{C}$ desde -10°C a $+65^{\circ}\text{C}$ con corrección por software. Su rango de operación va desde -40°C hasta $+85^{\circ}\text{C}$.



Figura 4-110: Sensor de temperatura ds1922l-f5# [45]

4.6.2.1.3 Usb (ds9490b#)

El DS9490B (Figura 4-111) es un adaptador USB en el cual se puede insertar un iButton tamaño F5. El iButton queda sujeto en su interior. Soporta el protocolo de comunicaciones 1-wire.



Figura 4-111: Usb (ds9490b#)

4.6.2.2 Metodología de toma de datos

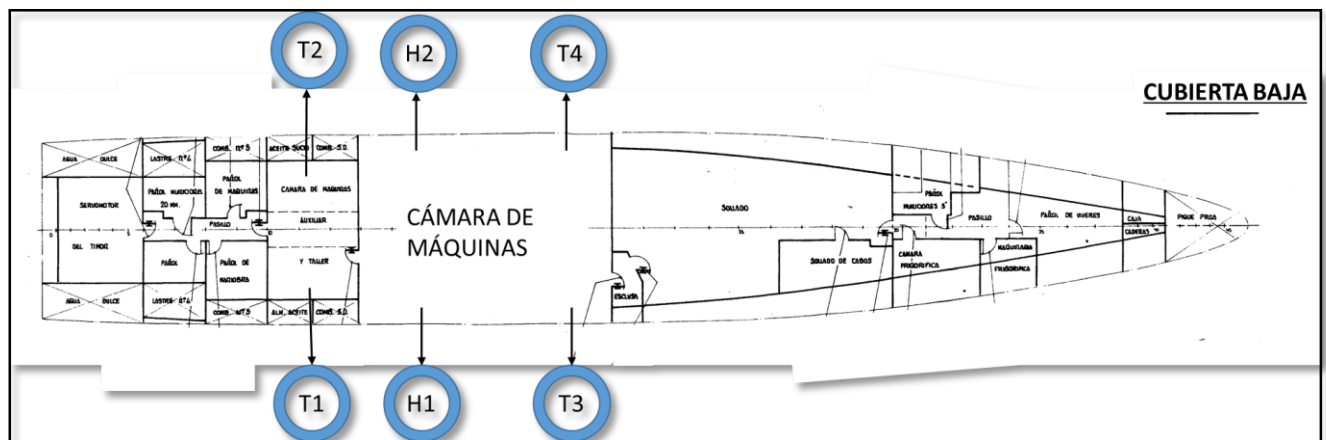
4.6.2.2.1 Zonas donde se ubicaron los sensores

La temperatura interna del buque que se emplea en la simulación, el patrullero Tabarca, es un factor importante a medir para conseguir asemejar lo más posible la simulación a las condiciones reales del buque. Para medir las temperaturas internas se utilizan sensores de temperatura y humedad.

Se decide instalar sensores de humedad y de temperatura en diferentes partes del Tabarca. Se distribuyen en el interior de la obra viva, obteniendo así mediciones de las temperaturas de confort y climatización a las que se encuentra dicho buque. Se coloca un total de cuatro (4) sensores de temperatura (Figura 4-110), colocados dos a cada banda de la sala de máquinas, y dos sensores de temperatura y humedad (Figura 4-109), uno a cada banda. Se colocan los sensores de la Figura 4-112

A photograph showing six circular samples arranged in two rows of three. The top row contains samples T1 (black), T2 (dark grey), T3 (white), and T4 (white). The bottom row contains samples H1 (dark grey) and H2 (white). Each sample has a teal circular label with its respective letter and number.

Se numeran con la letra T los de temperatura, siguiendo la distribución que emplea la Armada Española para numerar compartimentos a bordo, números pares a babor y números impares a estribor. También se comienza a numerar de popa hacia proa, por lo que se colocan según la distribución de la Figura 4-113:



85

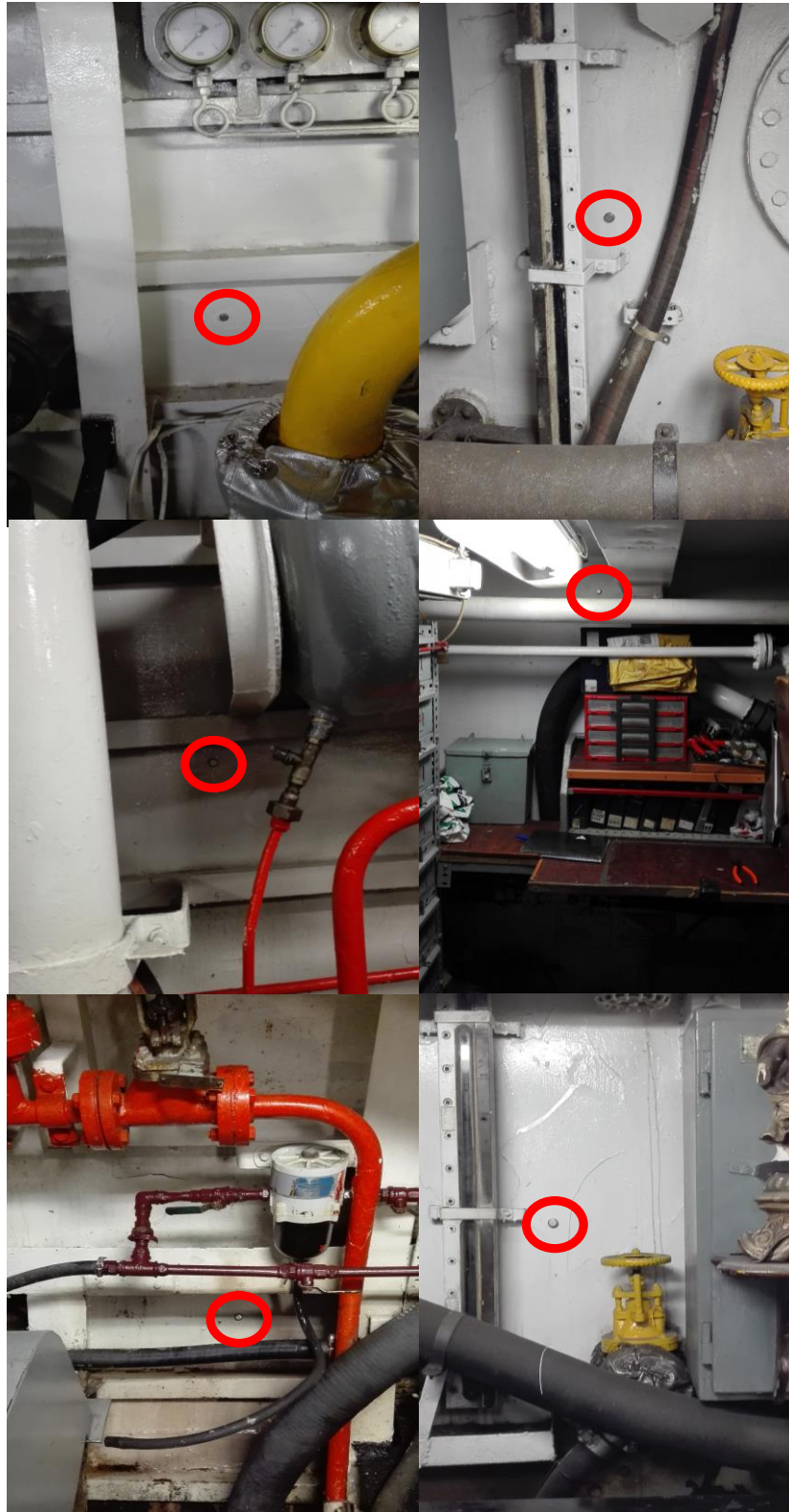


Figura 4-114: Ubicación de sensores a bordo del Tabarca

4.6.2.2.2 Protocolo de toma de datos

Se programan los sensores con ayuda del software *OneWireViewer* (Figura 4-115), donde podemos seleccionar la frecuencia de medición, el inicio de las mismas y la precisión que se desea medir.

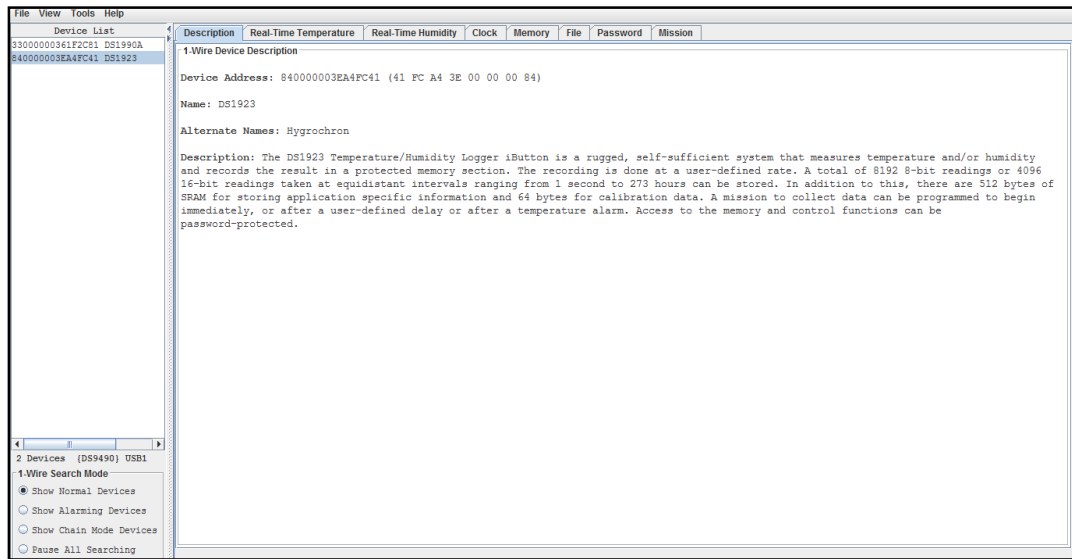


Figura 4-115: Software OneWireViewer

Se colocan los sensores el miércoles 27 de enero de 2016 por la mañana, y se recogen el martes 02 de febrero de 2016, permaneciendo a bordo del Tabarca realizando mediciones durante una semana, con lo que se pretende estudiar las diferencias de climatización entre días laborables en los que la dotación acude al buque, y los fines de semana en los que únicamente el personal de guardia se encuentra a bordo.

Se programan los sensores del Tabarca para que comiencen a medir a la misma hora, añadiendo retrasos para el inicio de las mediciones diferentes a cada sensor según la hora de programación, y de manera horaria durante los días que permanecen a bordo. Se programa la máxima precisión de los sensores para poder observar cualquier mínima diferencia (Figura 4-116).

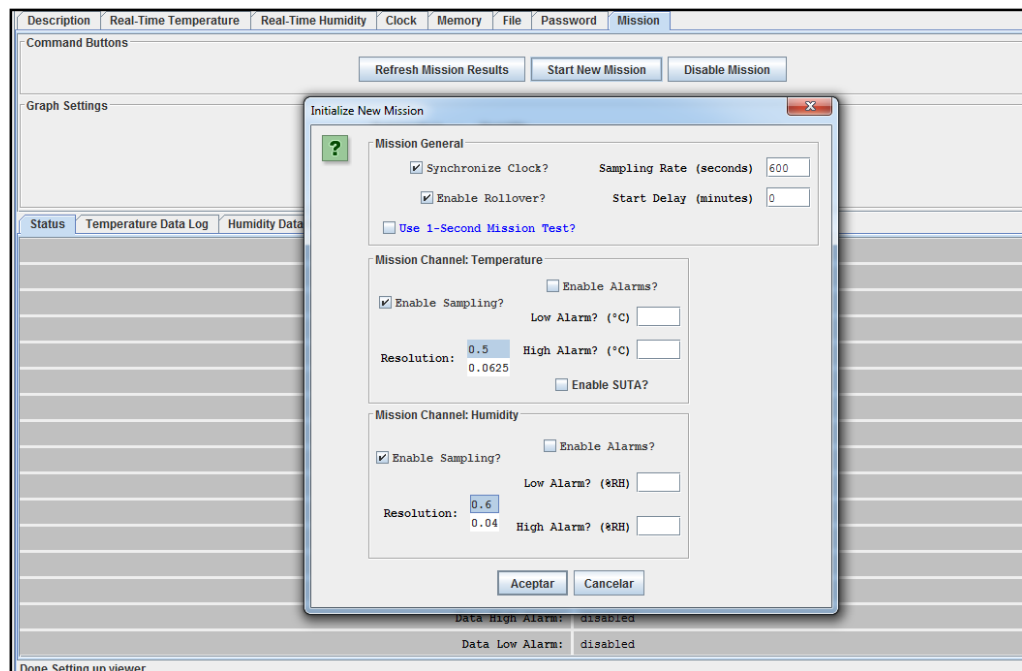


Figura 4-116: Programación de un sensor

Para realizar el análisis de datos una vez recogidos los sensores, se desprecian los datos del primer y último día para de esta manera obtener datos de días completos

4.6.2.3 Resultados

Con los datos de los diferentes sensores, se realiza el análisis obteniendo la temperatura media de los diferentes días en los que se realizan las mediciones.

Tabla 4-9: Temperatura media del buque por días

DIA	Sensor T1	Sensor T2	Sensor T3	Sensor T4	Sensor H1	Sensor H2
28/01/2016	14.273	13.718	13.145	12.913	13.737	13.666
29/01/2016	12.963	13.375	12.583	12.352	13.773	12.404
30/01/2016	12.807	12.445	12.584	12.295	13.995	12.385
31/01/2016	14.303	14.345	13.898	13.848	13.959	14.529
01/02/2016	14.873	14.309	13.579	13.495	14.302	14.730

Como se observa en la Tabla 4-9 no existe una diferencia clara entre las temperaturas medidas en la banda de estribor (Sensor T1, Sensor H1 y Sensor T3) y la banda de babor (Sensor T2, Sensor H2 y Sensor T4) por lo que se decide tomar como temperatura media diaria, la media de las temperaturas obtenidas por los diferentes sensores.

Tabla 4-10: Temperatura media diaria

DIA	TEMPERATURA MEDIA
28/01/2016	13.575
29/01/2016	12.908
30/01/2016	12.752
31/01/2016	14.147
01/02/2016	14.215

Con las temperaturas medias de cada día en el interior del Tabarca, se puede observar el pequeño incremento de temperatura el domingo 31 de enero, al permanecer durante todo el día anterior cerrado debido a que no se encontraba personal trabajando a bordo.

Con los datos de los sensores se estudia también la influencia de la hora en la temperatura, diferenciando entre horas de trabajo (aproximando de 08:00 a 16:00) en las que la dotación circula por el buque por lo que permanecen la mayor parte de los accesos abiertos, y horas a partir de finalización del horario laboral (16:00 a 08:00) cuando solo permanece el personal de guardia a bordo, y se ha aumentado la condición de estanqueidad del buque, por lo que los accesos se encontrarán cerrados.

Tabla 4-11: Temperaturas en función de la hora

DIA	TEMPERATURA DE 08:00 A 16:00	TEMPERATURA DE 16:00 A 08:00	DIFERENCIA DE TEMPERATURA
28/01/2016	14.198	14.318	0.120
29/01/2016	12.616	13.170	0.554
30/01/2016	12.620	12.920	0.300
31/01/2016	14.323	14.291	-0.032
01/02/2016	14.754	14.944	0.190

Ecuación 4-6: Variación de temperatura en tramos horarios

$$\Delta T^a = T_{16:00 \text{ A } 08:00} - T_{08:00 \text{ A } 16:00}$$

Se calcula la variación de temperatura entre los tramos horarios seleccionados, restando la temperatura en horario no laboral y la temperatura en horario laboral, observando que se produce un aumento de la temperatura en horario no laboral, con lo que se puede obtener la conclusión de que al permanecer los accesos al buque cerrados, se conserva de mejor manera la temperatura interna, con la excepción del día 31 de enero (señalado en negrita), donde se produce una mínima variación contraria al resto de los días medidos, pudiéndose deber ésta a una alta temperatura ambiente durante la mañana de ese día.

Finalmente tras el análisis completo de los datos, se calcula el promedio de las temperaturas medias de todos los días medidos (Tabla 4-10), siendo esta temperatura de 13.519 °C.

Para la simulación se empleará la máxima y la mínima temperatura registrada por los sensores para simular de la manera más exacta posible las condiciones reales. Estas temperaturas serán:

Tabla 4-12: Temperatura máxima y mínima alcanzada

Temperatura máxima	15.81 °C
Temperatura mínima	11.82 °C

4.6.3 Medición de la temperatura de agua de mar

4.6.3.1 Metodología de toma de datos

Para obtener la medición de la temperatura del mar en nuestra ubicación, se coloca un sensor de temperatura (Figura 4-110) en la dársena de la Escuela Naval Militar. Una vez obtenidos los datos se compararán con la temperatura proporcionada por Meteogalicia. Para poder ubicar el sensor en el agua, se hacen previamente unas pruebas que consisten en comprobar el hermetismo e impermeabilidad de las bolsas sumergibles “Aquapac” utilizadas para llevar objetos electrónicos en deportes de agua, ya que los sensores no deben entrar en contacto con el agua.

Se realizan varias comprobaciones, siendo la primera de ellas introducir gel de sílice (Figura 4-117) con indicador de color naranja (WG-2). El principal ingrediente es el dióxido de silicio y al no contener cloruro de cobalto lo convierte en un producto inofensivo y libre de ningún tipo de

contaminación, conservando los cambios de color según la cantidad de humedad absorbida que caracteriza al gel de sílice con indicador [46].



Figura 4-117: Gel de sílice con indicador de color naranja [46]

Principalmente se utiliza como absorbente con indicador de humedad en instrumentos de precisión, en medicina, en el proceso químico de la gasolina, en comestibles, ropa, piel, aparatos eléctricos, industria de gases y otros.

Una vez introducido el gel en las bolsas sumergible (izquierda) obtenemos resultados negativos, ya que al permanecer durante una noche sumergida en la dársena de la Escuela Naval Militar, el gel de sílice absorbe humedad y cambia de color pasando a ser color verde oscuro (derecha), lo que significa que la humedad relativa en el interior de la bolsa hermética es aproximadamente del 90%.

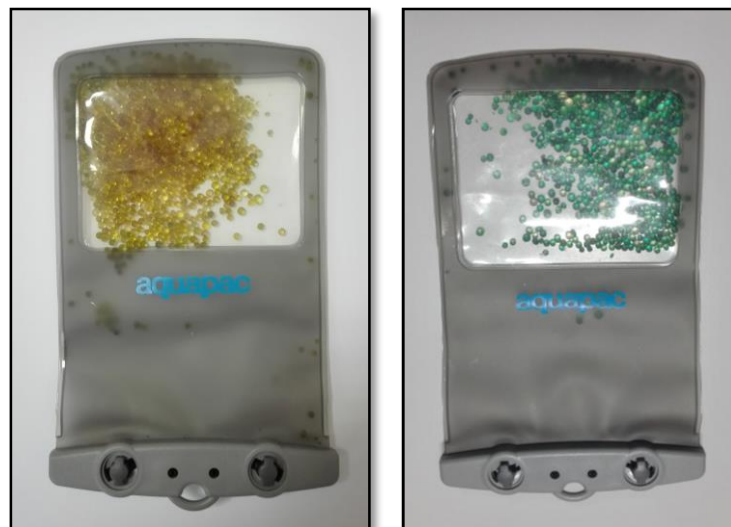


Figura 4-118: Bolsa sumergible con gel de sílice

Continuando con las pruebas, se decide introducir las bolsas sumergibles con papel (Figura 4-118) de folio en su interior, en caso de que entrara agua se mojaría el papel, y deberíamos buscar una alternativa. Se comprueba que no entra agua, y resultan ser bolsas sumergibles.

Como medida de seguridad, se emplea finalmente una Capsula Antiagua DS9107 (Figura 4-120) para iButton, que permite sumergir sensores iButton en el agua, protegiendo esta cápsula del contacto con el agua.

Se introduce el sensor en el interior, colocando la cara de medición (cara opuesta a la marcada para su identificación) en la parte izquierda de la Figura 4-119. Una vez introducido el sensor en la cápsula debe cerrarse correctamente enroscando ambas partes.



Figura 4-119: Cápsula Antiagua DS9107 con sensor



Figura 4-120: Cápsula Antiagua DS9107

Para poder medir la temperatura del mar en la dársena, se decide dejar el sensor con la capsula durante varios días sumergido, introduciendo la cápsula en una bolsa sumergible como medida extra de seguridad y para permitir mantenerlo sumergido a la profundidad y hacer posible su recogida (Figura 4-121).



Figura 4-121: Bolsa sumergible con cápsula en su interior

Para realizar las mediciones, se sumerge el sensor en el muelle Almirante Vierna de la Escuela Naval Militar de Marín el día 28 de enero hasta su recogida el día 10 de enero, teniendo así una amplia muestra de mediciones para realizar el estudio de manera más significativa.

4.6.3.2 Resultados y comparación con datos de MeteoGalicia

Se exportan los datos del software *OneWiseViewer* a Excel, donde se realiza el análisis de datos y la comparación con los datos obtenidos de la red de meteorología de MeteoGalicia.

Con el sensor de temperatura registramos la temperatura con intervalos de media hora, manteniendo las mediciones desde el día 28 de enero hasta el día 10 de febrero, días que descartamos por no obtener la información total del día.

Se calculan las medias de temperaturas diarias para estudiar de manera sencilla la coherencia de los datos, con el fin de poder desechar datos incoherentes.

Tabla 4-13: Temperaturas diarias medidas con sensores in situ

	Temperatura máxima	Temperatura media	Temperatura mínima
29-enero	13.62	12.17	10.30
30-enero	14.60	12.20	10.36
31-enero	14.60	13.72	12.81
01-febrero	13.81	13.27	12.62
02-febrero	13.94	13.24	12.31
03-febrero	13.68	12.67	10.17
04-febrero	14.60	13.68	12.81
05-febrero	14.75	11.79	9.11
06-febrero	14.94	13.60	10.86
07-febrero	14.60	12.73	9.48
08-febrero	13.94	13.58	13.12
09-febrero	13.81	13.32	12.24
MEDIA	14.24	13.00	11.35

Se observa en los resultados de las temperaturas medias, que la temperatura del mar sufre menos variaciones, manteniendo por norma general una temperatura estable en la media diaria.

Con los datos a emplear se crea un archivo de texto que incluirá los datos de la temperatura del mar medida junto al Tabarca cada media hora, desde las 00:00 horas del día 29 de enero hasta las 23:30 horas del día 9 de febrero, archivo que emplearemos como *External File* en el menú del type 9a de las simulaciones en *Trnsys*.

Como ejemplo de análisis se seleccionan los datos del día 29 de enero, realizando una gráfica de valores que se comparará con los datos obtenidos de la página web de MeteoGalicia (Figura 4-122 y Figura 4-123).



Figura 4-122: Datos de temperatura del mar de MeteoGalicia [6]

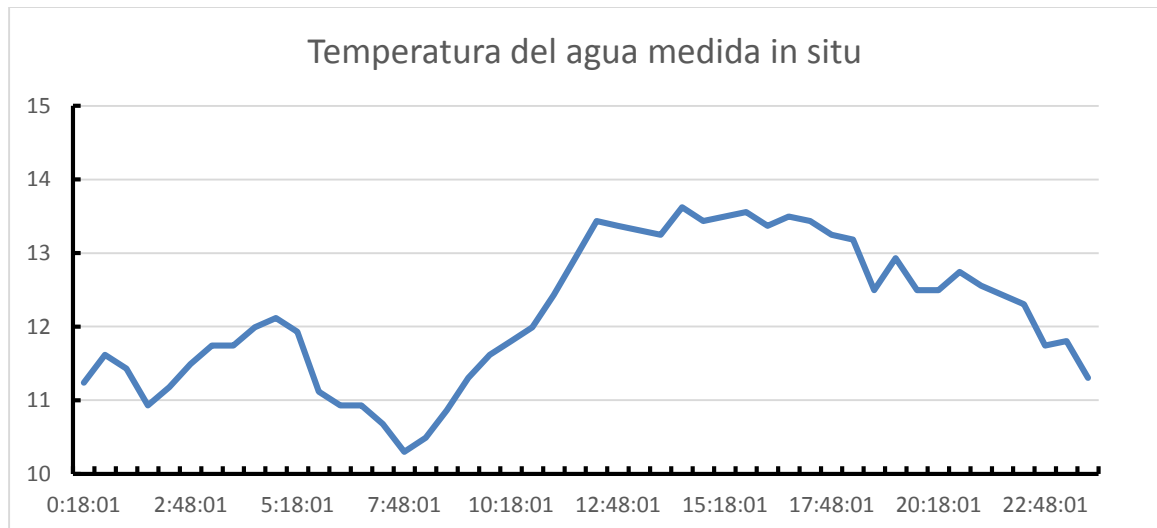


Figura 4-123: Datos de temperatura del mar de sensores

Se observa la diferencia significativa entre los datos obtenidos por medio de sensores de temperatura en la ubicación deseada, y las temperaturas de superficie de la página web de MeteoGalicia medidos en el puente de Rande (línea de color marrón).

Para el análisis de datos se comparan las temperaturas medias, máximas y mínimas de los días estudiados entre los datos medidos in situ y los datos ofrecidos por MeteoGalicia.

Tabla 4-14: Comparación de datos de temperatura del agua de mar

	SENSORES			METEOGALICIA		
	Temperatur a máxima	Temperatura media	Temperatur a mínima	Temperatur a máxima	Temperatur a media	Temperatur a mínima
29-enero	13.62	12.17	10.30	14.36	13.67	12.10
30-enero	14.60	12.22	10.36	14.65	14.10	12.91

31-enero	14.60	13.71	12.81	13.80	13.60	13.40
01-febrero	13.81	13.27	12.62	14.51	13.86	13.48
02-febrero	13.94	13.25	12.31	14.58	14.17	13.84
03-febrero	13.68	12.64	10.17	14.70	14.28	13.45
04-febrero	14.60	13.71	12.81	14.91	14.72	14.27
05-febrero	14.75	11.78	9.11	15.02	14.83	14.23
06-febrero	14.94	13.61	10.86	15.02	14.36	13.66
07-febrero	14.60	12.73	9.48	14.35	13.74	12.56
08-febrero	13.94	13.58	13.12	14.26	13.79	13.29
09-febrero	13.81	13.35	12.24	14.03	13.62	12.86

Para ello, se calcula el porcentaje de variación entre las dos posibles formas de obtener las temperaturas del agua de mar, el empleo de sensores de temperatura y los datos de MeteoGalicia.

Tabla 4-15: Porcentaje de diferencia entre datos de MeteoGalicia y sensores

	Temperatura máxima	Temperatura media	Temperatura mínima
29-enero	5.14	10.95	14.88
30-enero	0.34	13.33	19.74
31-enero	5.80	0.83	4.43
01-febrero	4.82	4.28	6.39
02-febrero	4.42	6.51	11.08
03-febrero	6.91	11.46	24.36
04-febrero	2.08	6.84	10.25
05-febrero	1.80	20.60	35.99
06-febrero	0.55	5.20	20.47
07-febrero	1.74	7.33	24.49
08-febrero	2.28	1.53	1.28
09-febrero	1.57	2.01	4.80
MEDIA	3.12	7.57	14.85

Como se puede observar en la tabla anterior, se obtiene un porcentaje de diferencia de las temperaturas medias medidas con los diferentes métodos de un 7.57%, valor que se considera influyente en las simulaciones en base a las simulaciones analizadas en el apartado 4.6.1.3.2, por lo que se emplearán los valores obtenidos por los sensores en la dársena de la Escuela Naval Militar.

5 SIMULACIÓN TÉRMICA A BORDO DEL PATRULLERO TABARCA

5.1 Descripción de la simulación

Se plantea la posibilidad de comparar los resultados de simulaciones realizadas con diferentes datos meteorológicos para analizar la influencia de las variables meteorológicas en las simulaciones a bordo de buques, como indica el título de este TFG.

Las simulaciones a realizar se diferenciarán únicamente en los archivos que introduciremos como datos en *Trnsys*, por ello se procede a describir el proceso de creación de la simulación del patrullero Tabarca.

Se comienza como en el resto de simulaciones con *SketchUp*, donde crearemos las diferentes zonas térmicas que representarán al patrullero Tabarca. Se decide emplear los datos obtenidos de las curvas hidrostáticas de la Clase Anaga, y los calados facilitados por la dotación del Tabarca (Geometría y volumen), con lo que se empleará un volumen de 350 toneladas para simular la obra viva del buque, y se decide emplear el mismo volumen para la obra muerta, ya que el objeto de análisis es el estudio de la influencia de las variables meteorológicas, puede aproximarse el patrullero a estas dos zonas térmicas para la simulación. Se emplea la orientación real del patrullero en sus estancias en puerto, para así poder emplear las temperaturas medidas a bordo manteniendo de manera más precisa la veracidad de esta simulación en cuanto a datos, siendo esta orientación Norte-Sur como se observa en la creación del modelo de *SketchUp* de la Figura 5-1.

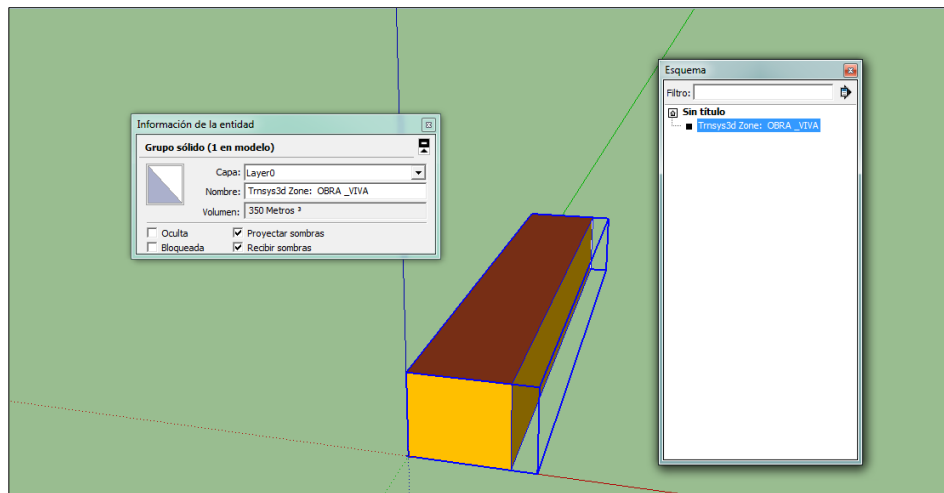


Figura 5-1: Modelo patrullero Tabarca en SketchUp

Se definen el tipo de paredes (Figura 5-2) de la obra viva, siendo todas ellas *Boundary_Wall* excepto la cara superior. Se definen *Boundary_Wall* ya que al estar en contacto con el agua de mar por estar sumergidas se puede introducir posteriormente en la simulación este input. La cara superior del volumen de obra viva se define como pared adyacente (*ADJ_WALL*) ya que estará en contacto con el volumen de la obra muerta. Para definir las paredes de la obra muerta, se definen todas excepto la cara inferior que se definirá como pared adyacente, como paredes externas (*EXT_WALL*) y así estar en contacto con el aire.

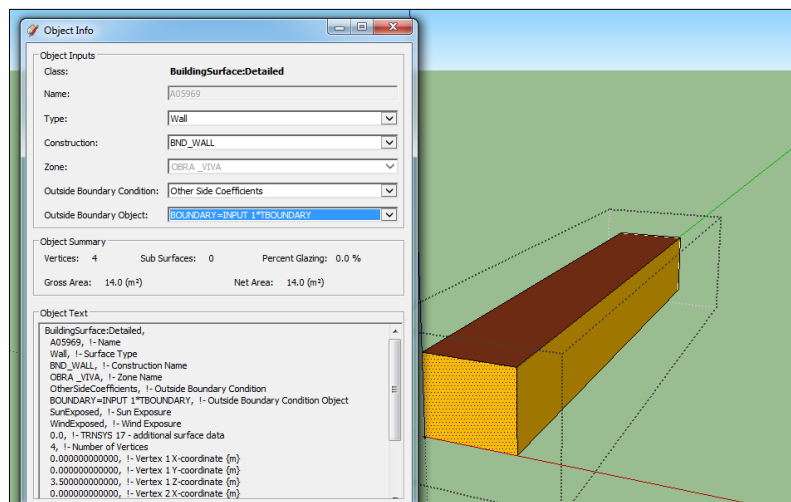


Figura 5-2: Definición de paredes en la obra viva

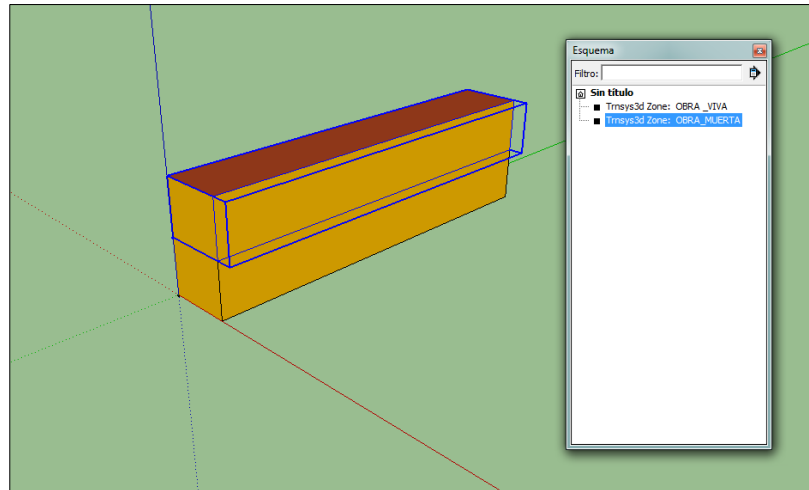


Figura 5-3: Modelo completo patrullero Tabarca en SketchUp

Se debe exportar a *Trnsys Simulation Studio* (Figura 5-4), una vez se haya creado el modelo de la simulación, creando un nuevo proyecto con los types por defecto. Se debe añadir principalmente dos types para la realización correcta de la simulación, siendo estos el type 9a y el type 65c.

El type 9a se empleará para introducir los datos de temperatura del mar a través de un archivo .txt que debe crearse previamente e introducirse en la pestaña *External File*. Este archivo se crea con los datos medidos por los sensores de temperatura con un tiempo de 30 minutos entre las tomas de los datos y durante varios días (datos analizados en el apartado Influencia de la temperatura del agua de mar en la simulación).

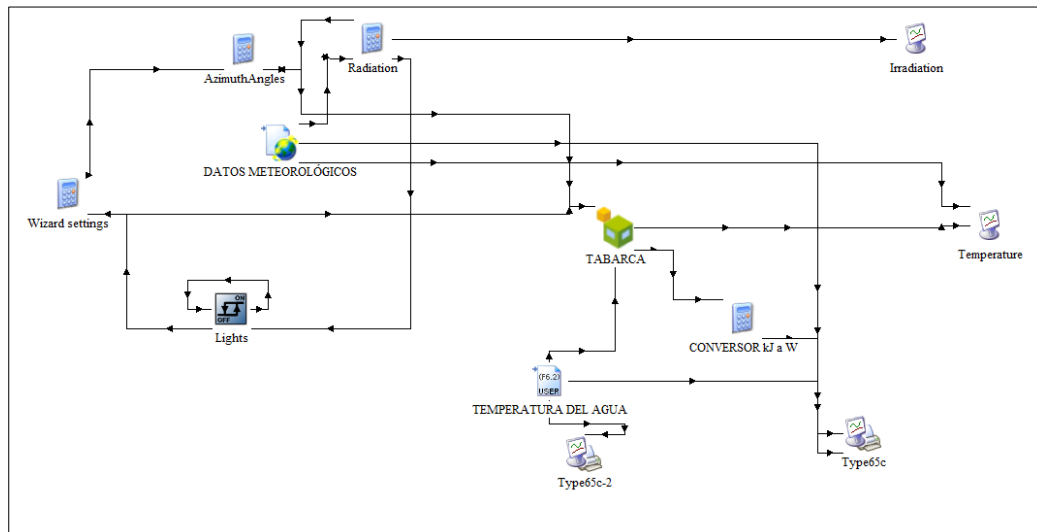
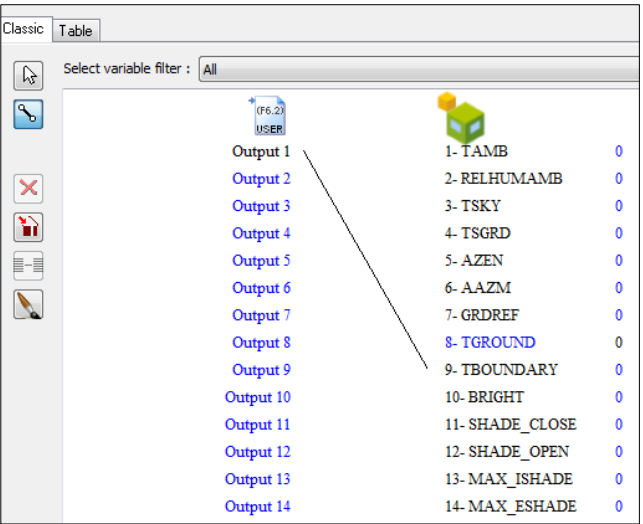


Figura 5-4: Simulación del patrullero Tabarca en Trnsys

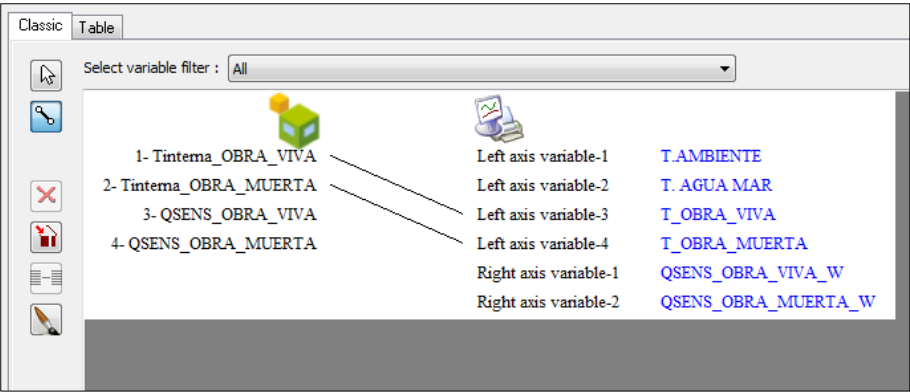
Se configura la relación entre el type 9a y el type Tabarca (type *Building* renombrado en esta simulación) enlazando el *Output 1* con *TBOUNDARY* como se puede observar en la Figura 5-5.



Output	Variable	Value
Output 1	1- TAMB	0
Output 2	2- RELHUMAMB	0
Output 3	3- TSKY	0
Output 4	4- TSGRD	0
Output 5	5- AZEN	0
Output 6	6- AAZM	0
Output 7	7- GRDREF	0
Output 8	8- TGROUND	0
Output 9	9- TBOUNDARY	0
Output 10	10- BRIGHT	0
Output 11	11- SHADE_CLOSE	0
Output 12	12- SHADE_OPEN	0
Output 13	13- MAX_ISHADE	0
Output 14	14- MAX_ESHADE	0

Figura 5-5: Relación entre type 9a y type Tabarca

Se añade el type 65c (Figura 5-6) para representar los resultados en gráficas y exportar los resultados de la simulación al archivo *External File* creado para la simulación. Se obtendrán como resultados en este type, los datos configurados como *Input* del mismo, siendo en la simulación las temperaturas internas de ambos volúmenes, temperatura ambiente y de agua de mar, y los intercambios de calor de los dos volúmenes.



Input	Variable
1- Tintema_OBRA_VIVA	Left axis variable-1 T.AMBIENTE
2- Tintema_OBRA_MUERTA	Left axis variable-2 T. AGUA MAR
3- QSENS_OBRA_VIVA	Left axis variable-3 T_OBRA_VIVA
4- QSENS_OBRA_MUERTA	Left axis variable-4 T_OBRA_MUERTA
	Right axis variable-1 QSENS_OBRA_VIVA_W
	Right axis variable-2 QSENS_OBRA_MUERTA_W

Figura 5-6: Inputs type 65c

Se debe añadir a mayores un convertor de unidades para obtener los datos de intercambios de calor en vatios (W) (Figura 5-7). Este convertor recibe como *Inputs* los intercambios de calor del type Tabarca y envía como *Output* al type de resultados, los datos medidos en vatios.

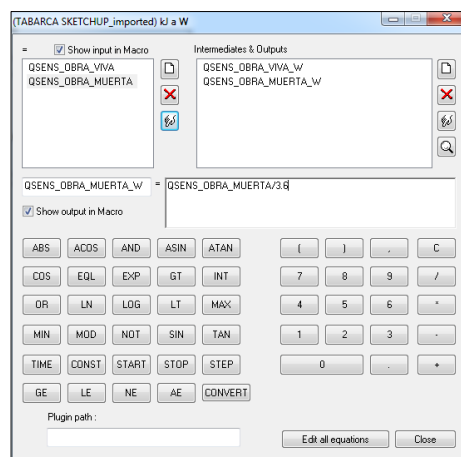


Figura 5-7: Conversor de unidades

Con todos los types que se necesitarán en la simulación una vez incluidos y relacionados, se debe configurar los tiempos de medición y las frecuencias de lecturas de datos (Figura 5-8), paso muy importante para obtener coherencia en la simulación. Se debe configurar en primer lugar la frecuencia de lectura de dato del type 9a para que introduzca datos de temperatura cada 30 minutos, para esto, se debe poner en el parámetro 4 de este type 0.5 horas.

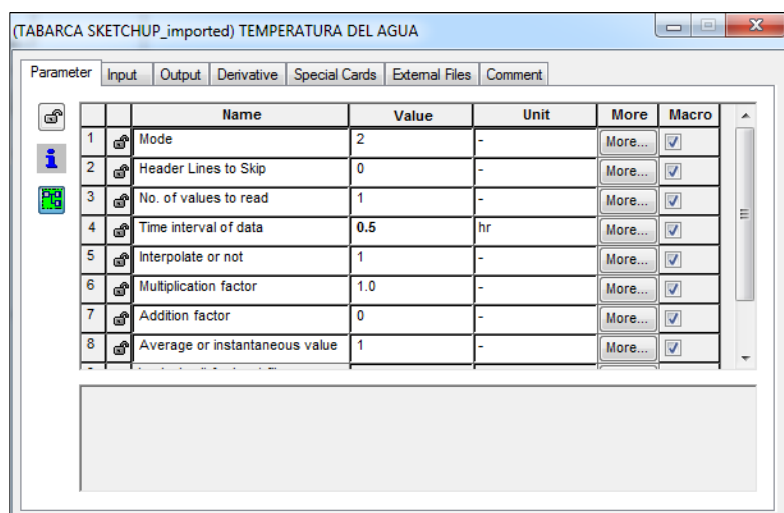


Figura 5-8: Configurar parámetros type 9a

Se debe fijar el tiempo de simulación, se usarán los días que se obtuvieron datos de la temperatura de agua de mar en la dársena de la Escuela Naval Militar con los sensores, por lo cual se debe simular desde el día 29 de enero al día 09 de febrero, lo que equivale en horas a las horas 672 y 960 (Anexo IV: Conversión de día y mes a hora) como se indica en la Figura 4-15: Selección del periodo de simulación.

Finalizado esto, se debe editar el modelo de la simulación, siendo en este caso el type Tabarca. Se asignará material a los cerramientos y se creará las condiciones de climatización.

Se asignan a los cerramientos los materiales y espesores más precisos de los disponibles en las librerías de *TRNBuild* (Figura 5-9) que se obtuvieron sobre el casco del Tabarca, siendo acero naval A y lana de roca, con los espesores definidos en la Tabla 5-1.

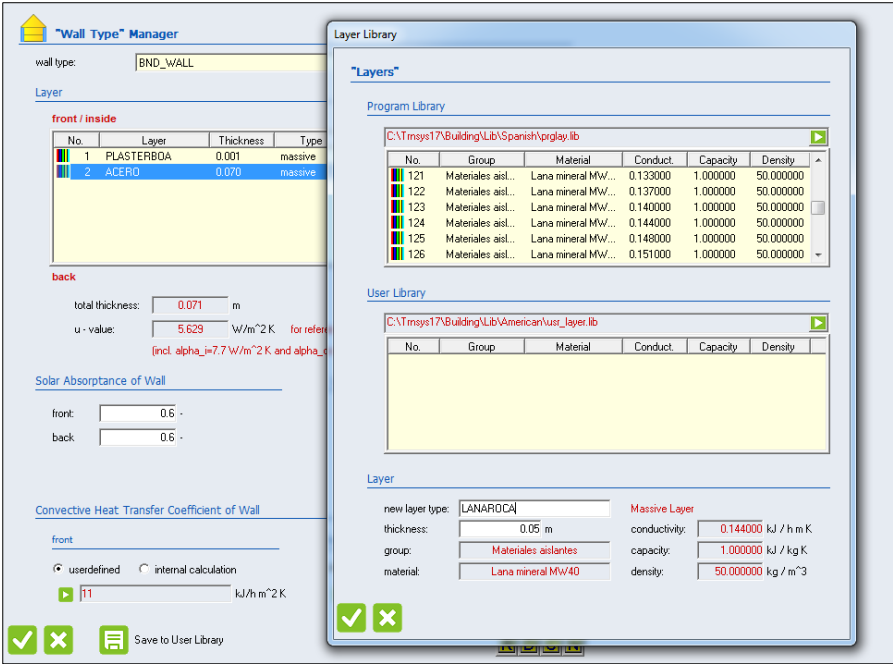


Figura 5-9: Asignar materiales a los cerramientos

En cuanto a la lana de roca, material aislante empleado principalmente en aislamiento térmico y como protección pasiva contra el fuego, se selecciona un valor medio de este material que oscila entre 0.05 y 0.031 W/m·K, por lo que se selecciona el material aislante con una conductividad de 0.0405 W/m·K lo que equivale en las librerías de *TRNBuild* a 0.144kJ/h·m·K.

Tabla 5-1: Materiales empleados en los cerramientos

Material	Espesor (metros)
Acero	0.07
Material aislante	0.05

Se utiliza el mismo tipo de cerramiento para todas las paredes de la simulación, ya que se supone que el material del casco será el material con mejores rendimientos térmicos, simulando así de manera más exacta los datos que se obtendrán en la obra viva.

Tabla 5-2: Propiedades de los materiales

Material	Conductividad kJ/h·m·K	Capacidad kJ/kg·K	Densidad kg/m3
Acero	180.00	0.50	7800
Lana mineral MW40	0.14	1.00	50.00

Se obtiene finalmente un espesor total de 0.120 metros y una transmitancia térmica de los cerramientos de 0.704 W/m²K (Figura 5-10).

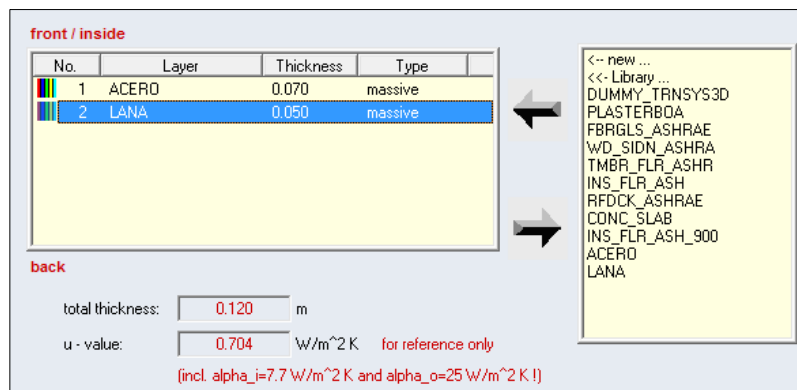


Figura 5-10: Propiedades de los cerramientos

Una vez editados los cerramientos, se debe configurar la climatización. Para determinar las temperaturas de consigna, se emplean los datos medidos con los sensores en la obra viva del Tabarca, seleccionando el valor máximo y el mínimo para establecer ese margen de temperaturas (Tabla 4-12: Temperatura máxima y mínima alcanzada), que se asemeja lo más posible a las condiciones reales.

Para esto se crean un sistema de calefacción programando la temperatura de 11.82 ° C (Figura 5-11) y un sistema de refrigeración programado a 15.81 ° C. Se debe activar la climatización en los dos volúmenes.

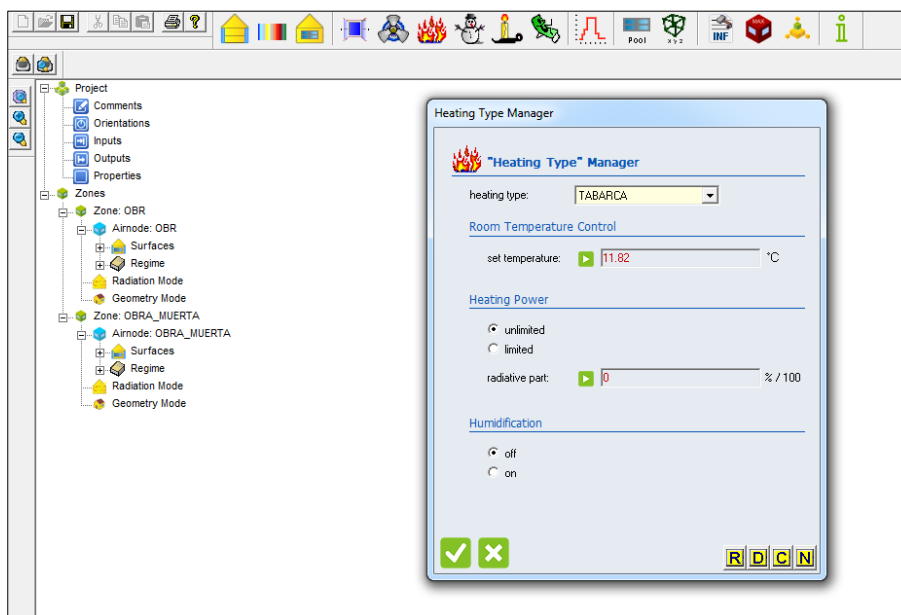


Figura 5-11: Climatización de los volúmenes

Por último, antes de ejecutar la simulación, se debe introducir en el type de Datos Meteorológicos, los datos obtenidos en *Meteonorm* de la ubicación más próxima al patrullero Tabarca, que es Pontevedra, con mediciones horarias. Se emplearán también los datos de temperatura del agua de los sensores colocados en la dársena. Como ya tenemos las horas de simulación designadas, solo sería necesario correr la primera simulación.

Una vez finalizada ésta y habiendo obtenido todos los resultados, se crea un nuevo archivo para los nuevos resultados, esta vez se usarán los datos de temperatura del agua de mar de los sensores y los datos obtenidos para la ubicación exacta del patrullero Tabarca por parte del equipo del CUVI, y correr nuevamente la simulación, habiendo de esta forma realizado dos simulaciones en las que únicamente difiere el archivo meteorológico introducido.

Para introducir los datos meteorológicos del equipo de investigación GTE de la Universidad de Vigo se debe realizar los pasos descritos en Esquema de procedimiento de construcción de archivos.

5.2 Resultados de las simulaciones

Se deben comparar los resultados obtenidos en las dos simulaciones, para ello empleamos los archivos de texto creados con los datos numéricos, y empleando las herramientas de Excel, comparamos las medidas medias de cada uno de los días en los que se hace la simulación.

Se diferencian la simulación en la que se emplean los datos de *Meteonorm* y la que emplea los datos del equipo de investigación GTE de la Universidad de Vigo.

En ambas simulaciones se analizan como resultados los datos de temperatura ambiente y temperatura del agua, temperaturas e intercambios de calor de la obra viva y de la obra muerta.

Tabla 5-3: Resultados de simulación con datos de Meteonorm y datos de agua de mar obtenidos con los sensores

	Temperatura ambiente (° C)	Temperatura de agua de mar (° C)	Temperatura interna obra viva (° C)	Temperatura interna obra muerta (° C)	Necesidad energética obra viva (W)	Necesidad energética obra muerta (W)
29-ene	11.52	12.17	15.94	15.94	2648.64	2296.54
30-ene	12.55	12.20	15.81	15.53	395.71	89.34
31-ene	13.78	13.72	15.65	14.65	0.00	0.00
01-feb	12.90	13.27	15.36	13.92	0.00	0.00
02-feb	12.69	13.24	15.04	13.25	0.00	0.00
03-feb	11.41	12.67	14.79	12.64	0.00	0.00
04-feb	7.21	13.68	14.38	11.93	0.00	-258.45
05-feb	10.06	11.79	13.78	11.82	0.00	-889.82
06-feb	9.82	13.60	13.51	11.82	0.00	-874.09
07-feb	10.81	12.73	13.27	11.82	0.00	-901.24
08-feb	8.34	13.58	13.20	11.82	0.00	-834.56
09-feb	7.71	13.32	13.11	11.82	0.00	-1100.21
Media	10.73	13.00	14.49	13.08	253.70	-206.04

Se observa en la tabla de resultados de la primera simulación (Tabla 5-3) los datos medios para cada uno de los días. En primer lugar se analiza las temperaturas ambiente medias de estos días, donde se puede observar que se produce una bajada de las temperaturas según pasan los días, teniendo los días 02 al 09 de febrero temperaturas inferiores a las mediciones de los primeros días. Estos datos son

obtenidos de la base de datos de *Meteonorm*, que realiza los promedios de los 10 últimos años, por lo que las temperaturas ambiente pueden tener cierto desfase con los datos reales de este año.

La temperatura del agua de mar permanecerá constante en las dos simulaciones ya que son datos medidos por los sensores de temperatura en la dársela de la ENM.

Analizando los datos de temperaturas internas de los dos volúmenes y los intercambios de calor, se observa que se mantiene una temperatura más elevada en la obra viva que en la obra muerta. Esto se puede deber al medio con el que se encuentra en contacto cada volumen, siendo la obra viva la que se mantiene en contacto con el agua de mar, que tiene temperaturas mayor y más uniformes que el aire, y la obra muerta la expuesta al aire a temperatura ambiente. Si se observa la media de temperatura del mar de los días estudiados, y la temperatura interna de la obra viva, se puede observar una proximidad entre resultados, siendo las dos mayores que las temperaturas de la obra muerta y del aire, en su mayor parte debido a que no interfieren de manera significativa otras variables meteorológicas como la radiación solar por estar sumergido en agua. Esta proximidad de resultados se debe también a que la temperatura media del agua de mar se encuentra entre las temperaturas de confort fijadas, y esto produce que el intercambio de calor, es decir, el aporte de calor necesario en esta zona para mantener las temperaturas de consigna sea nulo, existiendo únicamente los primeros días un intercambio de calor de signo positivo, lo que se traduce en la aplicación de refrigeración para bajar la temperatura interna a los límites establecidos. Por otra parte, la temperatura interna media de la obra muerta, es inferior a la temperatura de la obra viva, debiéndose esto a estar en contacto con el aire cuya temperatura ambiente media es de 10.73 ° C. Analizando los intercambios de calor de la obra muerta, se observa los dos primeros días el funcionamiento de la refrigeración para mantener las temperaturas internas entre los límites establecidos, al igual que pasaba en la obra viva. En los días 04 al 09 de febrero se observa un aporte de calor en la obra muerta para mantener las temperaturas en el límite inferior de la establecida, debiéndose esto a la bajada de la temperatura ambiente durante estos días.

Tabla 5-4: Resultados de simulación con datos del equipo de investigación GTE de la Universidad de Vigo y datos de agua de mar obtenidos con los sensores

	Temperatura ambiente (° C)	Temperatura de agua de mar (° C)	Temperatura interna obra viva (° C)	Temperatura interna obra muerta (° C)	Intercambio de calor obra viva (W)	Intercambio de calor obra muerta (W)
29-ene	11.90	12.17	15.94	15.94	2531.33	2217.70
30-ene	13.18	12.20	15.74	15.43	235.14	60.12
31-ene	9.02	13.72	15.21	14.10	0.00	0.00
01-feb	6.95	13.27	14.58	12.42	0.00	-0.55
02-feb	10.03	13.24	14.08	11.82	0.00	-458.74
03-feb	8.78	12.67	13.67	11.82	0.00	-863.38
04-feb	5.40	13.68	13.23	11.82	0.00	-1310.80
05-feb	5.27	11.79	12.82	11.82	0.00	-1520.21
06-feb	5.51	13.60	12.56	11.82	0.00	-1582.89

07-feb	6.39	12.73	12.36	11.82	0.00	-1551.01
08-feb	7.07	13.58	12.28	11.82	0.00	-1434.58
09-feb	9.23	13.32	12.45	11.82	0.00	-1072.21
Media	8.23	13.00	13.74	12.70	230.54	-626.38

En la segunda simulación, empleando los datos de temperatura del agua de mar y los datos meteorológicos proporcionados por el equipo de investigación GTE de la Universidad de Vigo para la ubicación exacta del patrullero Tabarca. Se observa una diferencia significativa en las temperatura ambiente con respecto a la simulación anterior, siendo esta más parecida a los datos reales de este año, debido al proceso de creación de estos archivos, siendo este proceso en empleo de técnicas de interpolación de datos con kriging. En cuanto a las temperaturas internas de la obra viva, siguen manteniendo una proximidad de valores con la temperatura del agua de mar y a pesar de ser inferiores las temperaturas internas de la obra viva, en esta simulación se observa que otras variables meteorológicas sí que afectan pero como se comenta en la tabla anterior. En cuanto a la obra muerta, se produce un aumento del aporte de calor comparando con la simulación anterior para mantener las temperaturas internas entre los límites establecidos, debido a la bajada de las temperaturas ambiente con los datos proporcionados por el equipo de investigación GTE de la Universidad de Vigo.

Tabla 5-5: Diferencias entre diferentes fuentes de datos meteorológicos

	Temperatura ambiente (%)	Temperatura interna obra viva (%)	Temperatura interna obra muerta (%)	Intercambio de calor obra viva (%)	Intercambio de calor obra muerta (%)
29-ene	-3.30	0.00	0.00	4.43	3.43
30-ene	-5.05	0.45	0.67	40.58	32.71
31-ene	34.53	2.84	3.74	0	0
01-feb	46.17	5.11	10.78	0	0
02-feb	20.92	6.39	10.82	0	0
03-feb	23.10	7.59	6.49	0	0
04-feb	25.09	7.98	0.94	0	80.28
05-feb	47.60	6.98	0.00	0	41.47
06-feb	43.87	7.04	0.00	0	44.78
07-feb	40.90	6.84	0.00	0	41.89
08-feb	15.26	6.96	0.00	0	41.83
09-feb	-19.63	5.06	0.00	0	-2.61
Media	23.00	5.27	2.79	22.50	35.47

La variación de los datos de temperatura ambiente es un 23%, producido por las diferentes formas de obtener la información en las diferentes simulaciones. Analizando las diferencias entre las

temperaturas internas, se observa que son bastante inferiores a la diferencia en la temperatura ambiente. Se produce un 5.27% de diferencia en la obra viva, y un 2.79% en la obra muerta, de lo que se puede deducir que la variación de temperaturas del ambiente afecta en mayor grado a la obra viva. Esto se explica de la siguiente manera, las temperaturas del agua y las temperaturas del ambiente mantienen una cierta diferencia, y la disminución de las temperaturas del ambiente afecta en mayor medida a la obra viva que a la obra muerta. Esto se debe probablemente, a los cerramientos empleados en esta simulación, considerando ambas zonas con las mismas características.

En cuanto a la demanda energética, se produce mayor diferencia en la obra muerta, ya que será necesario aportar más calor por la bajada de las temperaturas del ambiente para mantener las temperaturas límite establecidas.

6 CONCLUSIONES Y LÍNEAS FUTURAS

En el presente apartado se comentarán las conclusiones obtenidas durante el desarrollo y estudio del TFG, y se propondrán líneas de trabajo futuras para próximos años.

Se obtiene como conclusión general, que las variables meteorológicas influyen de manera considerable en las simulaciones térmicas, para lo cual se deben obtener datos meteorológicos medidos in situ evitando así alteraciones en los resultados. Se considera de gran influencia en estas simulaciones, los cerramientos empleados en las simulaciones, afectando esto de manera considerable en los resultados. Se debe realizar la simulación con los cerramientos reales en la obra muerta, para de esta manera obtener la influencia real de las variables meteorológicas en buques, ya que los resultados obtenidos en el apartado anterior no se consideran representativos ni veraces al no realizarse la simulación con las características reales de la obra muerta.

6.1 Buque

En el presente trabajo se han utilizado los datos conocidos del patrullero Tabarca, empleando en determinadas circunstancias datos asumidos para la simplificación de la simulación.

En primer lugar, se ha empleado un volumen de obra viva obtenido con las Curvas Hidrostáticas de la Clase Anaga, sin estudiar o analizar posibles modificaciones en dichas curvas en el caso concreto del Patrullero de estudio debido a grandes obras o reparaciones, lo que puede provocar alteraciones en los resultados comparándolos con los datos reales.

No se tienen en cuenta las ganancias térmicas por ocupación de cada zona en cada momento, simplificándose esto en diferenciación en el análisis de temperaturas entre horario laboral, con todo el personal a bordo, y horario no laboral, donde solo el personal de guardia se encuentra a bordo.

Por otra parte, en cuanto a la simulación y el software empleado, se define el buque como una plataforma estática que permanece siempre en la misma ubicación como si se tratase de una edificación ya que el software empleado es de simulaciones térmicas en edificaciones. No se considera la posibilidad de que el buque realice una navegación, para lo cual se emplean los datos de temperatura obtenidos por medio de los sensores durante una semana en la que permanece en puerto base.

Para finalizar con el buque, se emplea un volumen de obra viva fijo, lo cual puede variar en determinadas circunstancias, como por ejemplo, los días en los que se realiza petróleo o carga de víveres. Esto se realiza para simplificación de cálculos y reducir la variación de la simulación cada periodo corto de tiempo.

6.2 Datos meteorológicos

Durante el desarrollo del presente trabajo, se emplean gran cantidad de datos meteorológicos obtenidos de diferentes formas.

En primer lugar, se emplean sensores de temperatura para realizar mediciones y monitorizar las temperaturas internas de la obra viva del buque y la temperatura de agua de mar en la dársena de la Escuela Naval Militar, para lo cual se toma un periodo de tiempo como generalización de los datos (del día 29 de enero al 10 de febrero) sin conocer realmente si este periodo de tiempo se considera anómalo observando el conjunto anual de datos. Además, como se comenta en el apartado anterior, no se monitoriza la temperatura en la obra muerta, lo cual produce una alteración de la simulación.

Para la obtención de datos de temperatura del mar, se utiliza la ubicación próxima a la que se considera ubicación normal del patrullero para sus estancias en puerto, pudiendo verse alterados los datos medidos en la dársena por la frecuente actividad de diferentes embarcaciones en la dársena, o por la propia forma de la dársena, con lo que pueden obtenerse datos de temperaturas del agua de mar no representativas de la zona de la Ría de Pontevedra, debido a que pueden influir posibles corrientes frías o cálidas en las temperaturas, dato que no se ve representado en los datos obtenidos.

En cuanto a la radiación, no se consideran sombras en el buque estudiado, produciéndose de esta manera diferencias con la realidad. Los edificios cercanos, como el Isaac Peral, producen sombras proyectadas sobre el buque en determinadas horas del día, por lo que variarían los resultados de la simulación en la obra muerta principalmente (como se observa en la Figura 6-1 y Figura 6-2). La radiación también variaría en caso de encontrarse el buque navegando, ya que variará la posición respecto al sol.



Figura 6-1: Posición del Tabarca respecto a los edificios próximos



Figura 6-2: Posición del Tabarca respecto a los edificios próximos

6.3 Estación meteorológica

Se monta y programa una estación meteorológica en la terraza del Centro Universitario de la Defensa, pretendiendo obtener datos reales y poder compararlos con los datos facilitados por el CUVI. Se ubica en el CUD por la necesidad de disponer de una red Wi-Fi para el envío de datos, lo cual produce una alteración en los posibles resultados a obtener respecto a la idea original de colocar la estación a bordo del Patrullero Tabarca para obtener así datos lo más precisos y reales posibles.

Se instala en primer momento empleando los conductos del sistema de climatización del CUD para la colocación de la caja dentro de la que se medirá la temperatura (Figura 6-3). Estos datos se consideran despreciables e inutilizables debido al funcionamiento de la climatización del CUD, lo que afecta de manera notable a las mediciones. Influye también en estas alteraciones, la disposición de los codos de PVC colocados en cada extremo de la caja, lo cual se hace para obtener una circulación del aire, pero al colocarse la entrada/salida de la caja en posición vertical como se observa en la Figura 6-3, asciende el calor radiado por los conductos de climatización.



Figura 6-3: Primera colocación de la estación meteorológica



Figura 6-4: Segunda colocación de la estación meteorológica

Para intentar solventar este problema se ubica en el suelo de la terraza, elevándolo 40 centímetros del suelo (Figura 6-4). A pesar de esto, se continúa produciendo alteraciones en las mediciones debido al calentamiento de la caja por su continua exposición a la radiación solar. También se producen lecturas erróneas en cuanto a la dirección e intensidad del viento por estar próximo a un muro que protege de vientos de componente Sur y Oeste. Se instala un sistema de recirculación de aire, consistiendo este en un pequeño ventilador en uno de los orificios laterales donde están los codos de PVC. Se consiguen datos utilizables durante el corto periodo en el que una pila de 12V mantiene el ventilador en funcionamiento.

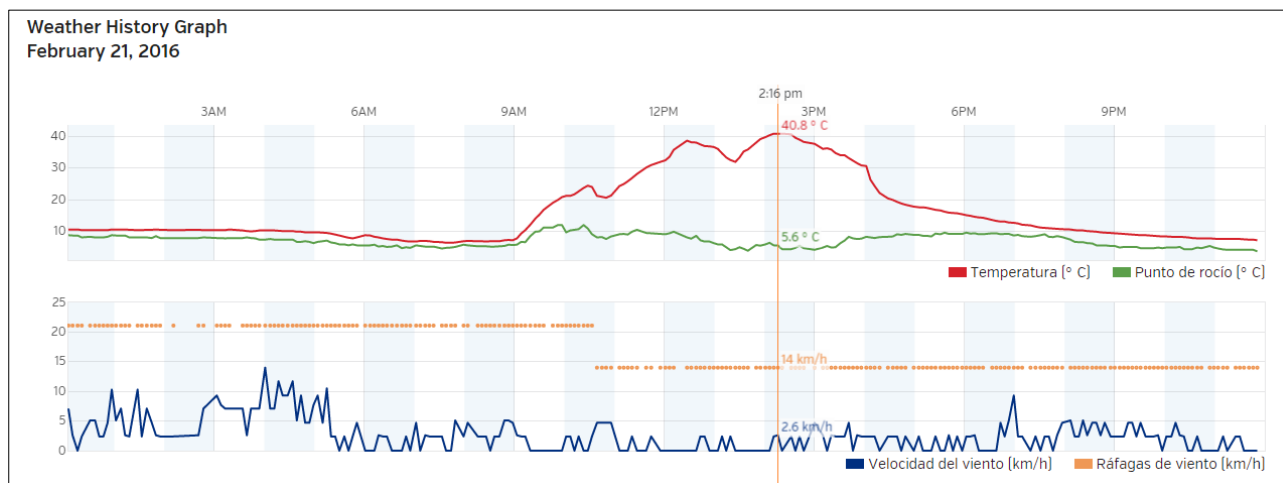


Figura 6-5: Datos obtenidos por la estación meteorológica

Se puede observar en la Figura 6-5 que se producen picos de temperatura e hasta 40.8 °C lo cual permite comprobar el error en la toma de datos.

En el presente trabajo no se consigue realizar una simulación en el Tabarca con los datos medidos in situ, debido a las mediciones inservibles obtenidas.

6.4 Simulación en Trnsys

En las simulaciones térmicas con *Trnsys*, se han empleado materiales de las librerías de *Trnsys* para definir los cerramientos, empleando acero con características lo más próximas posibles a los datos de aceros navales, además de materiales aislantes con valores medios, sin conocer el caso concreto de nuestro objeto de estudio. Además se emplea el mismo cerramiento para definir la obra viva y la obra muerta, esto se hace debido a que no se miden ni poseen datos de temperaturas de la obra muerta, aparte de no conocer las posibles infiltraciones en esta zona, por lo que se considera emplear los mismos datos obtenidos con los sensores de temperatura en ambas zonas, y por ello asignar el mismo

material. No se considera en cuanto a materiales, las capas de pintura existentes al considerarse de poca influencia y no conocerse las características concretas de estas.

En cuanto a geometría, se consideran dos volúmenes iguales como modelo del Tabarca, representando uno de ellos la obra viva y el otro la obra muerta. En ambos volúmenes se considera un volumen de 350 toneladas, lo cual es real en el caso concreto en el que se miden los calados del patrullero y se emplean las Curvas Hidrostáticas disponibles, pero se considera el mismo volumen en la obra muerta, lo cual no representa la realidad. Esto se debe a que se busca analizar la influencia de las variables meteorológicas y no el comportamiento del buque, para lo que se simplifica el modelo representando dos volúmenes iguales y con los mismos materiales.

7 LÍNEAS FUTURAS

Se exponen como futuras líneas de trabajo para años próximos las siguientes:

7.1 Validación del modelo y simulación en *Trnsys*

Se propone como futuras líneas de trabajo, el análisis y consideración de infiltraciones en el buque, así como el estudio de los materiales de los cerramientos de la obra muerta, pretendiendo así, obtener resultados de las influencias meteorológicas en buques con mayor precisión y exactitud.

Se propone también el estudio de la ocupación del buque los diferentes días de la semana, permitiendo así obtener el modelo energético del buque de estudio y sus posibles mejoras en rendimiento y consumos energéticos.

En cuanto al punto de atraque, se propone el estudio de cambiar su ubicación en largas estancias en puerto, analizando si algún punto de la dársena de la ENM resultaría más conveniente para conseguir mayor ahorro energético.

A nivel interno del buque y su dotación, se propone concienciarles de los posibles fallos que repercuten a la demanda energética, tal cómo, mantener una mala estanqueidad en el buque, lo que hace que sea necesario el empleo de calefacción para el mantenimiento de las temperaturas deseadas a bordo.

7.2 Datos meteorológicos y estación

En cuanto a futuros estudios de los datos meteorológicos, se propone la calibración de la estación meteorológica creada durante el presente Trabajo de Fin de Grado y la correcta ubicación en un lugar idóneo, debiendo solucionar para ello, la necesidad de ubicarla en un lugar con una red Wi-Fi disponible para el envío de datos y lo más próxima posible al buque de estudio. Se debe buscar también un lugar más próximo al buque de estudio, considerando que no se deben producir lecturas erróneas por influencias de muros, paredes o alteraciones de los datos de temperatura y viento por otros factores.

Una vez calibrada y ubicada la estación, se propone la obtención de datos meteorológicos con los que crear un archivo que será empleado posteriormente en las simulaciones de *Trnsys*, pudiendo así analizar las diferencias entre los datos medidos *in situ* y los facilitados por la Universidad de Vigo.

8 BIBLIOGRAFÍA

- [1] (2016). *Instituto tecnológico de Galicia*. Available: <http://www.itg.es/>.
- [2] R. Clemente Alfonso, "Diseño de una cubierta solar fotovoltaica conectada a la red eléctrica e integrada en un edificio industrial," 2006.
- [3] M. Bhandari, S. Shrestha and J. New, "Evaluation of weather datasets for building energy simulation," *Energy Build.*, vol. 49, pp. 109-118, 2012.
- [4] U. DOE, "Getting started with EnergyPlus," 2009.
- [5] (24/02/2016). *Organización Meteorológica Mundial*. Available: <http://www.wmo.int/youth/es/prediccion-meteorologica>.
- [6] (24/02/2016). *MeteoGalicia. Junta de Galicia. Consejería de Medio Ambiente y Ordenación del Territorio*. Available: <http://www.meteogalicia.es/web/index.action>.
- [7] S. S. Casado and A. L. González, "Datos obtidos nas estacións meteorolóxicas, predición meteorolóxica: aplicación na aula," *Boletín Das Ciencias*, pp. 133-133, 2014.
- [8] (). *Google Maps*.
- [9] (23/02/2016). *Pontevedra Viva*. Available: <http://pontevedraviva.com/xeral/16105/olor-pontevedra-sensores-universidade-vigo-satelite-aire-agua-ruido-convenio-investigacion/?lang=es>.
- [10] (02/03/2016). *Calificación Energética de Edificios*. Available: <http://magaralingeneria.com/calificacion-energetica-de-edificios>.
- [11] S. F. Larsen and G. Lesino, "Modelo térmico del programa SIMEDIF de simulación de edificios," *Energías Renovables Y Medio Ambiente*, vol. 9, pp. 15-24, 2001.
- [12] P. L. Llamas, "Eficiencia energética y medio ambiente," *Información Comercial Española, ICE: Revista De Economía*, pp. 75-92, 2009.

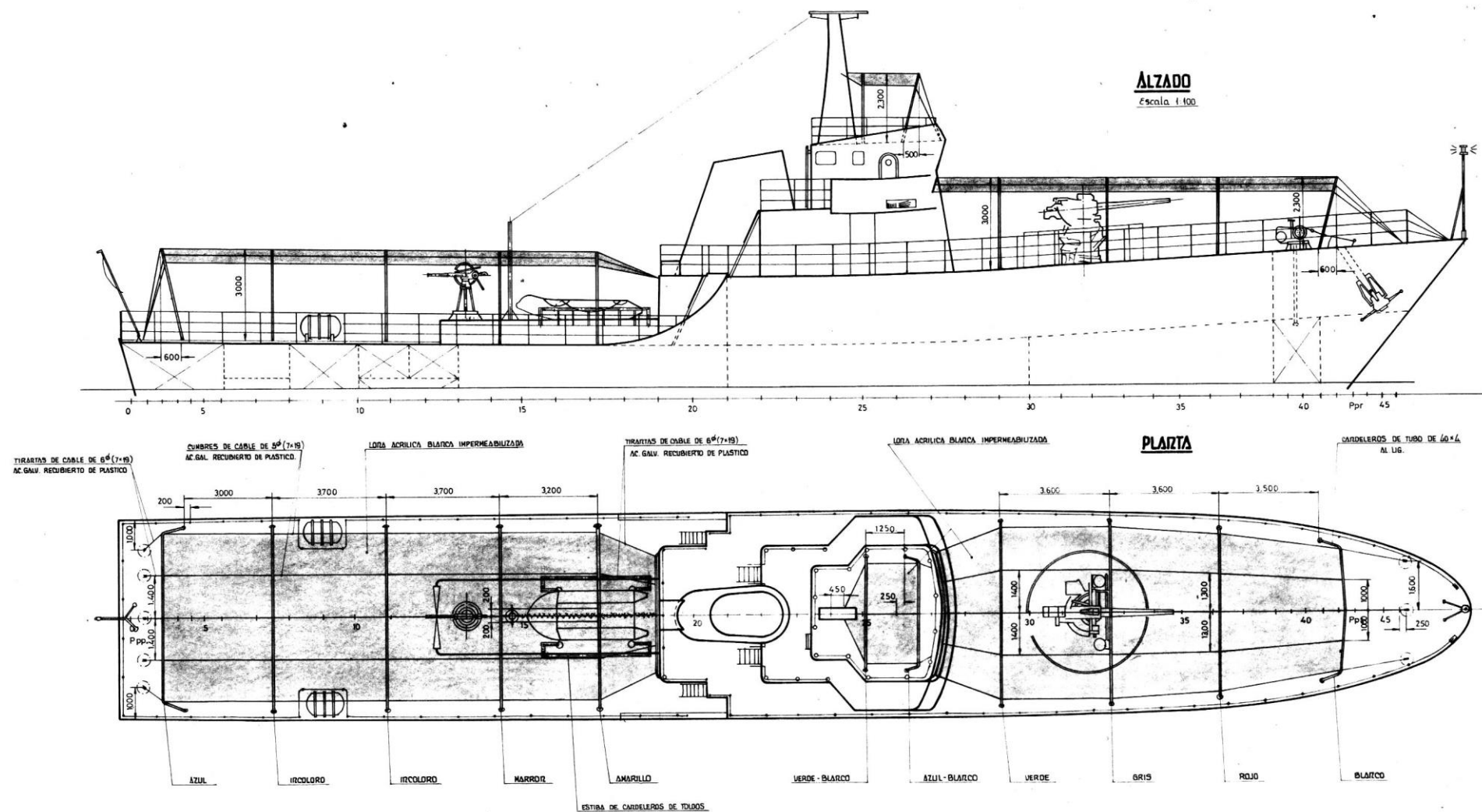
- [13] (29/02/2016). *Simulación energética de edificios*. Available: http://www.itg.es/?page_id=528.
- [14] C. Ulloa, P. Eguía, J. L. Miguez, J. Porteiro, J. M. Pousada-Carballo and A. Cacabelos, "Feasibility of using a Stirling engine-based micro-CHP to provide heat and electricity to a recreational sailing boat in different European ports," *Appl. Therm. Eng.*, vol. 59, pp. 414-424, 2013.
- [15] E. Fridell, H. Winnes and L. Styhre, "Medidas para mejorar la eficiencia energética en el transporte marítimo," 2013.
- [16] M. M. González, V. D. Casás, F. L. Peña and R. J. Duro, "Buques de Pesca y Eficiencia Energética: Proyecto Peixe Verde Actividad del Grupo Integrado de Ingeniería," *Ingeniería Naval*, pp. 94-107, 2008.
- [17] F. M. Carrascal Villar, *Estudio De Las Necesidades Energéticas De Un Buque En Diferentes Ubicaciones*, 2015.
- [18] J. TÉCNICA, "PROCEDIMIENTOS Y ASPECTOS DE LA SIMULACIÓN DE INSTALACIONES TÉRMICAS EN EDIFICIOS." *Asociación Técnica Española De Climatización Y Refrigeración (ATECYR) Para El Instituto Para La Diversificación Y Ahorro De La Energía (IDAE)*, 2008.
- [19] D. J. Stine, *Google SketchUp 8 for Interior Designers*. SDC Publications, 2012.
- [20] S. Klein, "TRNSYS 17 manual: a TRaNsient SYstem Simulation program," *Solar Energy Laboratory.University of Wisconsin, Madison: WI, USA*, 2012.
- [21] P. EGUÍA OLLER, J. M. LÓPEZ SUÁREZ, A. QUICLER COSTAS, L. M. LÓPEZ OCHOA, "Simulación de edificios con Trnsys17," vol. 1, 2012.
- [22] A. R. Yarre, "La utilizacion del programa Meteonorm para generar la base de datos meteorologicos que requiere la adaptacion del Energy-10 para la Argentina," .
- [23] (10/01/2016). *Meteonorm Datos climáticos mundiales*. Available: <http://store.aiguasol.coop/castellano/meteonorm-v7.html>.
- [24] (). *SparkFun Weather Station*. Available: <https://www.sparkfun.com/products/12757>.
- [25] Unidad Didáctica "Meteorología y Climatología", *Unidad Didáctica "Meteorología Y Climatología"*. Semana de la Ciencia y la Tecnología 2004: 2004.
- [26] I. Z. López and E. C. del Arco, *Meteorología Y Climatología*. Editorial UNED, 2010.
- [27] A. D. DE EDUCACIÓN, "Revista de Metodología de Ciencias Sociales," *EMPIRIA.Revista De Metodología De Ciencias Sociales*, pp. 101-132, 2010.
- [28] P. A. Tipler and G. Mosca, *Física Para La Ciencia Y La Tecnología*. Reverté, 2005.
- [29] J.J. Natividad Alvarado, R. Espinoza Paredes, E. Ramos Martínez, "Tratamiento de Datos Meteorológicos para la Simulación Energética," 2012.

- [30] (). *Radiación Solar*. Available: http://www.aemet.es/documentos/es/eltiempo/observacion/radiacion/Radiacion_Solar.pdf.
- [31] (). *Armada Española*. Available: http://armada.mde.es/ArmadaPortal/page/Portal/ArmadaEspañola/_inicio_home/prefLang_es/.
- [32] (). *Buques Armada Española*. Available: <http://www.buquesdeguerra.com/es/buques-armada-espanola/patrulleros/patrullero-clase-anaga.html>.
- [33] B. Givoni, *Passive Low Energy Cooling of Buildings*. John Wiley & Sons, 1994.
- [34] M. E. Arce Fariña, Cacabelos Reyes, Antón, Ulloa Sande, Carlos and González Gil, Arturo y Rodríguez Rodríguez, Javier, "Análisis del consumo energético en buques con estancias prolongadas en puerto," in 2015, .
- [35] J. Bierman and E. Kincanon, "Reconsidering Archimedes' principle," *The Physics Teacher*, vol. 41, pp. 340-344, 2003.
- [36] J. Olivella Puig, *Teoría Del Buque. Flotabilidad Y Estabilidad. Problemas*. Universitat Politècnica de Catalunya. Iniciativa Digital Politècnica, 2004.
- [37] (10/02/2016). *Salvamento Marítimo*.
- [38] F. MOREU CURBERA and E. Martínez Jiménez, "_Astronomía y Navegación_", *Ed.San José.Vigo*, 1968.
- [39] I. Velez-Pareja, "Master Consultores 2009," *Analisis De Sensibilidad*, 2009.
- [40] L. T. Blank, A. J. Tarquin and Carlos Freddy Mendoza B., *Ingeniería Económica*. McGraw-Hill, 1991.
- [41] (28/01/2016). *Propiedades térmicas de los materiales*. Available: <http://www.miliarium.com/Prontuario/Tablas/Quimica/PropiedadesTermicas.asp>.
- [42] Código Técnico de la Edificación, "Documento Básico Ahorro de Energía," *Secretaría De*, 2006.
- [43] F. Bre, V. D. Fachinotti and G. Bearzot, "Mecánica Computacional, Volume XXXII. Number 37. Heat and Mass Transfer (A)," .
- [44] (). *iButton*. Available: <http://ibutton.cl/>.
- [45] (09/01/2016). **Grupo Haiyang Yinhai España**. Available: <http://www.geldesilice.com/es/productos/productos-de-gel-de-silicesilicagel/gel-de-silice-con-indicador-de-color-naranja-wg-2/27/10/5>.

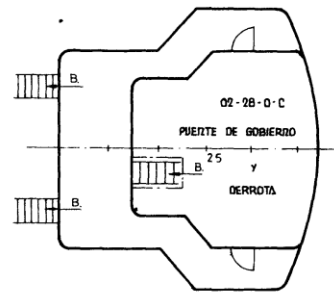
ANEXO I: ESCALA DE BEAUFORT

Número de Beaufort	Velocidad del viento (km/h)	Nudos (millas náuticas/h)	Denominación	Aspecto del mar	Efectos en tierra
0	0 a 1	< 1	Calma	Despejado	Calma, el humo asciende verticalmente
1	2 a 5	1 a 3	Ventolina	Pequeñas olas, pero sin espuma	El humo indica la dirección del viento
2	6 a 11	4 a 6	Flojito (Brisa muy débil)	Crestas de apariencia vítrea, sin romper	Se mueven las hojas de los árboles, empiezan a moverse los molinos
3	12 a 19	7 a 10	Flojo (Brisa débil)	Pequeñas olas, crestas rompientes.	Se agitan las hojas, ondulan las banderas
4	20 a 28	11 a 16	Bonancible (Brisa moderada)	Borreguillos numerosos, olas cada vez más largas	Se levanta polvo y papeles, se agitan las copas de los árboles
5	29 a 38	17 a 21	Fresquito (Brisa fresca)	Olas medianas y alargadas, borreguillos muy abundantes	Pequeños movimientos de los árboles, superficie de los lagos ondulada
6	39 a 49	22 a 27	Fresco (Brisa fuerte)	Comienzan a formarse olas grandes, crestas rompientes, espuma	Se mueven las ramas de los árboles, dificultad para mantener abierto el paraguas.
7	50 a 61	28 a 33	Frescachón (Viento fuerte)	Mar gruesa, con espuma arrastrada en dirección del viento	Se mueven los árboles grandes, dificultad para andar contra el viento
8	62 a 74	34 a 40	Temporal (Viento duro)	Grandes olas rompientes, franjas de espuma	Se quiebran las copas de los árboles, circulación de personas dificultosa
9	75 a 88	41 a 47	Temporal fuerte (Muy duro)	Olas muy grandes, rompientes. Visibilidad mermada	Daños en árboles, imposible andar contra el viento
10	89 a 102	48 a 55	Temporal duro (Temporal)	Olas muy gruesas con crestas empenachadas. Superficie del mar blanca.	Árboles arrancados, daños en la estructura de las construcciones
11	103 a 117	56 a 63	Temporal muy duro (Borrasca)	Olas excepcionalmente grandes, mar completamente blanca, visibilidad muy reducida	Estragos abundantes en construcciones, tejados y árboles

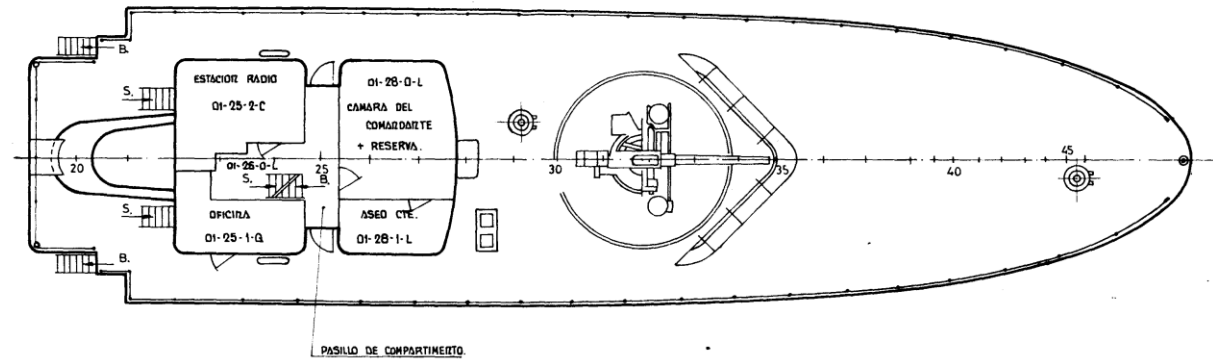
ANEXO II: PLANOS PATRULLERO TABARCA



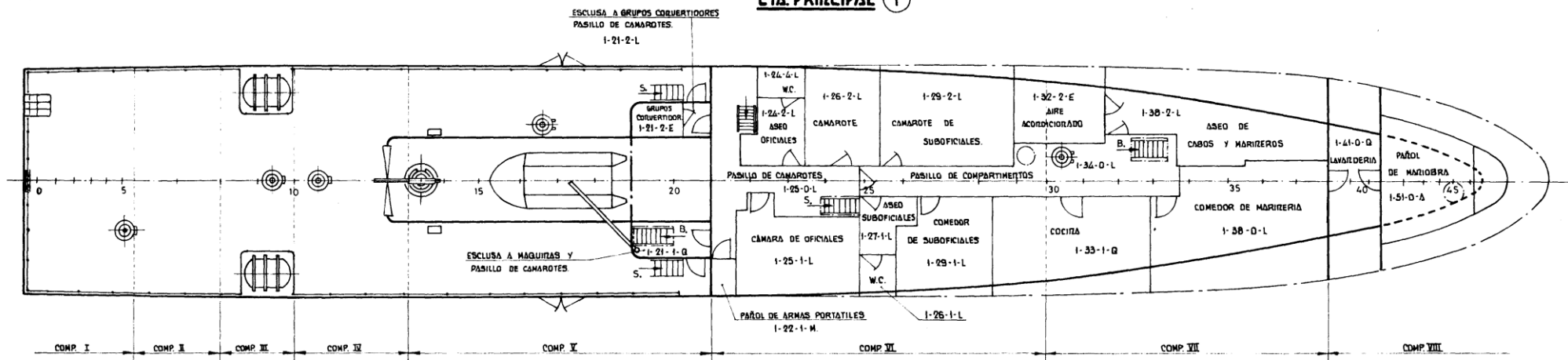
CTA. PUENTE 02



CTA. CASTILLO 01

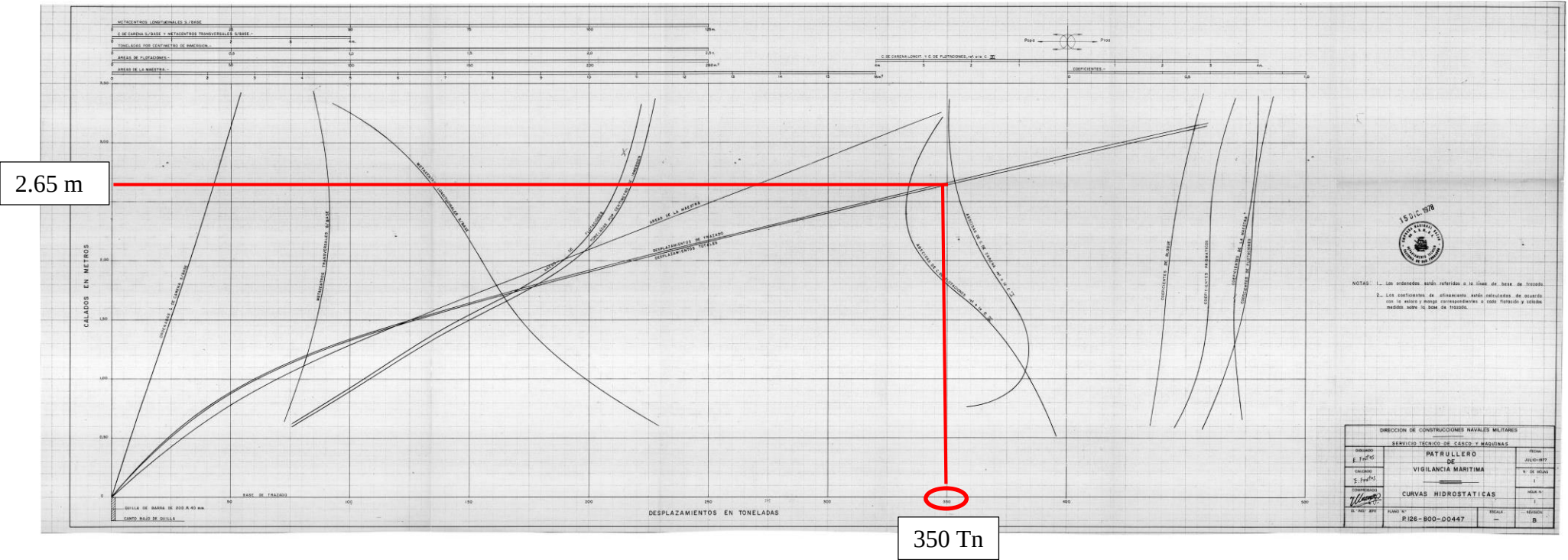


CTA. PRINCIPAL 1





ANEXO III: CURVAS HIDROSTÁTICAS PATRULLERO TABARCA



ANEXO IV: CONVERSIÓN DE DÍA Y MES A HORA

ENERO

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
H	24	48	72	96	120	144	168	192	216	240	264	288	312	336	360	384	408	432	456	480	504	528	552	576	600	624	648	672	696	720	744

FEBRERO

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
H	768	792	816	840	864	888	912	936	960	984	1008	1032	1056	1080	1104	1128	1152	1176	1200	1224	1248	1272	1296	1320	1344	1368	1392	1416

MARZO

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
H	1440	1464	1488	1512	1536	1560	1584	1608	1632	1656	1680	1704	1728	1752	1776	1800	1824	1848	1872	1896	1920	1944	1968	1992	2016	2040	2064	2088	2112	2136	2160

ABRIL

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
H	2184	2208	2232	2256	2280	2304	2328	2352	2376	2400	2424	2448	2472	2496	2520	2544	2568	2592	2616	2640	2664	2688	2712	2736	2760	2784	2808	2832	2856	2880

MAYO

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
H	2904	2928	2952	2976	3000	3024	3048	3072	3096	3120	3144	3168	3192	3216	3240	3264	3288	3312	3336	3360	3384	3408	3432	3456	3480	3504	3528	3552	3576	3600	3624

JUNIO

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
H	3648	3672	3696	3720	3744	3768	3792	3816	3840	3864	3888	3912	3936	3960	3984	4008	4032	4056	4080	4104	4128	4152	4176	4200	4224	4248	4272	4296	4320	4344

JULIO

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
H	4368	4392	4416	4440	4464	4488	4512	4536	4560	4584	4608	4632	4656	4680	4704	4728	4752	4776	4800	4824	4848	4872	4896	4920	4944	4968	4992	5016	5040	5064	5088

AGOSTO

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
H	5112	5136	5160	5184	5208	5232	5256	5280	5304	5328	5352	5376	5400	5424	5448	5472	5496	5520	5544	5568	5592	5616	5640	5664	5688	5712	5736	5760	5784	5808	5832

SEPTIEMBRE

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
H	5856	5880	5904	5928	5952	5976	6000	6024	6048	6072	6096	6120	6144	6168	6192	6216	6240	6264	6288	6312	6336	6360	6384	6408	6432	6456	6480	6504	6528	6552

OCTUBRE

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
H	6576	6600	6624	6648	6672	6696	6720	6744	6768	6792	6816	6840	6864	6888	6912	6936	6960	6984	7008	7032	7056	7080	7104	7128	7152	7176	7200	7224	7248	7272	7296

NOVIEMBRE

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
H	7320	7344	7368	7392	7416	7440	7464	7488	7512	7536	7560	7584	7608	7632	7656	7680	7704	7728	7752	7776	7800	7824	7848	7872	7896	7920	7944	7968	7992	8016

DICIEMBRE

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
H	8040	8064	8088	8112	8136	8160	8184	8208	8232	8256	8280	8304	8328	8352	8376	8400	8424	8448	8472	8496	8520	8544	8568	8592	8616	8640	8664	8688	8712	8736	8760

ANEXO V: CÓDIGO ARDUINO

```
/*
Weather Station using the Electric Imp
By: Nathan Seidle
SparkFun Electronics
Date: October 4th, 2013

License: This code is public domain but you buy me a beer if you use this and we
meet someday (Beerware license).

Much of this is based on Mike Grusin's USB Weather Board code.

This code reads all the various sensors (wind speed, direction, rain gauge,
humidty, pressure, light, batt_lvl)
and sends it to the imp, which then forwards that data to an Imp Agent on the
cloud that does some processing then
bounces the weather data to Wunderground.

The Imp Shield has Card Detect tied to pin A0. We use A0 for wind direction. You
will need to cut the trace on the Imp shield.

Current:
130 for 2 seconds while transmitting
~30mA during sleep

Todo:
Reset after 45 days to avoid millis roll over problems

What was the wind direction and speed gust for the last 10 minutes?
Is the 3.3V pin tied on the weather shield or elsewhere?
*/

#include <avr/wdt.h> //We need watch dog for this program
#include <Wire.h> //I2C needed for sensors
#include "MPL3115A2.h" //Pressure sensor
#include "HTU21D.h" //Humidity sensor

//#define ENABLE_LIGHTNING

//SoftwareSerial imp(8, 9); // RX, TX into Imp pin 7
```

```

MPL3115A2 myPressure; //Create an instance of the pressure sensor
HTU21D myHumidity; //Create an instance of the humidity sensor

//Hardware pin definitions
//-----
// digital I/O pins
const byte WSPEED = 3;
const byte RAIN = 2;
const byte STAT1 = 7;

#ifdef ENABLE_LIGHTNING
const byte LIGHTNING_IRQ = 4; //Not really an interrupt pin, we will catch it in
software
const byte slaveSelectPin = 10; //SS for AS3935
#endif

// analog I/O pins
const byte WDIR = A0;
const byte LIGHT = A1;
const byte BATT = A2;
const byte REFERENCE_3V3 = A3;
//-----

#ifdef ENABLE_LIGHTNING
#include "AS3935.h" //Lighting dtector
#include <SPI.h> //Needed for lighting sensor

byte SPItransfer(byte sendByte);

AS3935 AS3935(SPItransfer, slaveSelectPin, LIGHTNING_IRQ);
#endif

//Global Variables
//-----
long lastSecond; //The millis counter to see when a second rolls by
unsigned int minutesSinceLastReset; //Used to reset variables after 24 hours. Imp
should tell us when it's midnight, this is backup.
byte seconds; //When it hits 60, increase the current minute
byte seconds_2m; //Keeps track of the "wind speed/dir avg" over last 2 minutes
array of data
byte minutes; //Keeps track of where we are in various arrays of data
byte minutes_10m; //Keeps track of where we are in wind gust/dir over last 10
minutes array of data

```

```
long lastWindCheck = 0;
volatile long lastWindIRQ = 0;
volatile byte windClicks = 0;

#ifdef ENABLE_LIGHTNING
byte lightning_distance = 0;
#endif

//We need to keep track of the following variables:
//Wind speed/dir each update (no storage)
//Wind gust/dir over the day (no storage)
//Wind speed/dir, avg over 2 minutes (store 1 per second)
//Wind gust/dir over last 10 minutes (store 1 per minute)
//Rain over the past hour (store 1 per minute)
//Total rain over date (store one per day)

byte windspdavg[120]; //120 bytes to keep track of 2 minute average
#define WIND_DIR_AVG_SIZE 120
int winddiravg[WIND_DIR_AVG_SIZE]; //120 ints to keep track of 2 minute average
float windgust_10m[10]; //10 floats to keep track of largest gust in the last 10
minutes
int windgustdirection_10m[10]; //10 ints to keep track of 10 minute max
volatile float rainHour[60]; //60 floating numbers to keep track of 60 minutes of
rain

//These are all the weather values that wunderground expects:
int winddir; // [0-360 instantaneous wind direction]
float windspeedmph; // [mph instantaneous wind speed]
float windgustmph; // [mph current wind gust, using software specific time period]
int windgustdir; // [0-360 using software specific time period]
float windspdmph_avg2m; // [mph 2 minute average wind speed mph]
int winddir_avg2m; // [0-360 2 minute average wind direction]
float windgustmph_10m; // [mph past 10 minutes wind gust mph ]
int windgustdir_10m; // [0-360 past 10 minutes wind gust direction]
float humidity; // [%]
float tempf; // [temperature F]
float rainin; // [rain inches over the past hour]] -- the accumulated rainfall in
the past 60 min
volatile float dailyrainin; // [rain inches so far today in local time]
//float baromin = 30.03; // [barom in] - It's hard to calculate baromin locally, do
this in the agent
```

```

float pressure;

//float dewptf; // [dewpoint F] - It's hard to calculate dewpoint locally, do this
in the agent

//These are not wunderground values, they are just for us
float batt_lvl = 11.8;
float light_lvl = 0.72;

// volatiles are subject to modification by IRQs
volatile unsigned long raintime, rainlast, raininterval, rain;

//-----

//Interrupt routines (these are called by the hardware interrupts, not by the main
code)
//-----
void rainIRQ()
// Count rain gauge bucket tips as they occur
// Activated by the magnet and reed switch in the rain gauge, attached to input D2
{
    raintime = millis(); // grab current time
    raininterval = raintime - rainlast; // calculate interval between this and last
event

    if (raininterval > 10) // ignore switch-bounce glitches less than 10mS after
initial edge
    {
        dailyrainin += 0.011; //Each dump is 0.011" of water
        rainHour[minutes] += 0.011; //Increase this minute's amount of rain

        rainlast = raintime; // set up for next event
    }
}

void wspeedIRQ()
// Activated by the magnet in the anemometer (2 ticks per rotation), attached to
input D3
{
    if (millis() - lastWindIRQ > 10) // Ignore switch-bounce glitches less than 10ms
(142MPH max reading) after the reed switch closes
    {
        lastWindIRQ = millis(); //Grab the current time
        windClicks++; //There is 1.492MPH for each click per second.
    }
}

```

```
}

void setup()
{
    wdt_reset(); //Pet the dog
    wdt_disable(); //We don't want the watchdog during init

    Serial.begin(9600);

    pinMode(WSPPEED, INPUT_PULLUP); // input from wind meters windspeed sensor
    pinMode(RAIN, INPUT_PULLUP); // input from wind meters rain gauge sensor

    pinMode(WDIR, INPUT);
    pinMode(LIGHT, INPUT);
    pinMode(BATT, INPUT);
    pinMode(REFERENCE_3V3, INPUT);

    pinMode(STAT1, OUTPUT);

    midnightReset(); //Reset rain totals

    //Configure the pressure sensor
    myPressure.begin(); // Get sensor online
    myPressure.setModeBarometer(); // Measure pressure in Pascals from 20 to 110 kPa
    myPressure.setOversampleRate(128); // Set Oversample to the recommended 128
    myPressure.enableEventFlags(); // Enable all three pressure and temp event flags
    myPressure.setModeActive(); // Go to active mode and start measuring!

    //Configure the humidity sensor
    myHumidity.begin();

#ifdef ENABLE_LIGHTNING
    startLightning(); //Init the lighting sensor
#endif

    seconds = 0;
    lastSecond = millis();

    // attach external interrupt pins to IRQ functions
    attachInterrupt(0, rainIRQ, FALLING);
```

```

attachInterrupt(1, wspeedIRQ, FALLING);

// turn on interrupts
interrupts();

Serial.println("Wimp Weather Station online!");
reportWeather();

// wdt_enable(WDTO_1S); //Unleash the beast
}

void loop()
{
  wdt_reset(); //Pet the dog

  //Keep track of which minute it is
  if(millis() - lastSecond >= 1000)
  {
    lastSecond += 1000;

    //Take a speed and direction reading every second for 2 minute average
    if(++seconds_2m > 119) seconds_2m = 0;

    //Calc the wind speed and direction every second for 120 second to get 2
minute average
    windspeedmph = get_wind_speed();
    winddir = get_wind_direction();
    windspdavg[seconds_2m] = (int)windspeedmph;
    winddiravg[seconds_2m] = winddir;
    //if(seconds_2m % 10 == 0) displayArrays();

    //Check to see if this is a gust for the minute
    if(windspeedmph > windgust_10m[minutes_10m])
    {
      windgust_10m[minutes_10m] = windspeedmph;
      windgustdirection_10m[minutes_10m] = winddir;
    }

    //Check to see if this is a gust for the day
    //Resets at midnight each night
    if(windspeedmph > windgustmph)
    {

```

```
        windgustmph = windspeedmph;
        windgustdir = winddir;
    }

    //Blink stat LED briefly to show we are alive
    digitalWrite(STAT1, HIGH);
    //reportWeather(); //Print the current readings. Takes 172ms.
    delay(25);
    digitalWrite(STAT1, LOW);

    //If we roll over 60 seconds then update the arrays for rain and windgust
    if(++seconds > 59)
    {
        seconds = 0;

        if(++minutes > 59) minutes = 0;
        if(++minutes_10m > 9) minutes_10m = 0;

        rainHour[minutes] = 0; //Zero out this minute's rainfall amount
        windgust_10m[minutes_10m] = 0; //Zero out this minute's gust

        minutesSinceLastReset++; //It's been another minute since last
night's midnight reset
    }
}

//Check to see if there's been lighting
#ifdef ENABLE_LIGHTNING
    if(digitalRead(LIGHTNING_IRQ) == HIGH)
    {
        //We've got something!
        lightning_distance = readLightning();
    }
#endif

//Wait for the imp to ping us with the ! character
if(Serial.available())
{
    byte incoming = Serial.read();
    if(incoming == '!')
```

```
{
    reportWeather(); //Send all the current readings out the imp and to
its agent for posting to wunderground. Takes 196ms
    //Serial.print("Pinged!");

#ifdef ENABLE_LIGHTNING
    //Give imp time to transmit then read any erroneous lightning strike
    delay(1000); //Give the Imp time to transmit
    readLightning(); //Clear any readings and forget it
#endif

}

else if(incoming == '@') //Special character from Imp indicating midnight
local time
{
    midnightReset(); //Reset a bunch of variables like rain and daily
total rain
    //Serial.print("Midnight reset");
}

else if(incoming == '#') //Special character from Imp indicating a hardware
reset
{
    //Serial.print("Watchdog reset");
    delay(5000); //This will cause the system to reset because we don't
pet the dog
}

}

//If we go for more than 24 hours without a midnight reset then force a reset
//24 hours * 60 mins/hr = 1,440 minutes + 10 extra minutes. We hope that Imp is
doing it.
if(minutesSinceLastReset > (1440 + 10))
{
    midnightReset(); //Reset a bunch of variables like rain and daily total
rain
    //Serial.print("Emergency midnight reset");
}

delay(100); //Update every 100ms. No need to go any faster.
}

//Prints the various arrays for debugging
void displayArrays()
{
```

```
//Windgusts in this hour
Serial.println();
Serial.print(minutes);
Serial.print(":");
Serial.println(seconds);

Serial.print("Windgust last 10 minutes:");
for(int i = 0 ; i < 10 ; i++)
{
    if(i % 10 == 0) Serial.println();
    Serial.print(" ");
    Serial.print(windgust_10m[i]);
}

//Wind speed avg for past 2 minutes
/*Serial.println();
Serial.print("Wind 2 min avg:");
for(int i = 0 ; i < 120 ; i++)
{
    if(i % 30 == 0) Serial.println();
    Serial.print(" ");
    Serial.print(windspdavg[i]);
}*/

//Rain for last hour
Serial.println();
Serial.print("Rain hour:");
for(int i = 0 ; i < 60 ; i++)
{
    if(i % 30 == 0) Serial.println();
    Serial.print(" ");
    Serial.print(rainHour[i]);
}

}

//When the imp tells us it's midnight, reset the total amount of rain and gusts
void midnightReset()
{
    dailyrainin = 0; //Reset daily amount of rain
```

```

    windgustmph = 0; //Zero out the windgust for the day
    windgustdir = 0; //Zero out the gust direction for the day

    minutes = 0; //Reset minute tracker
    seconds = 0;
    lastSecond = millis(); //Reset variable used to track minutes

    minutesSinceLastReset = 0; //Zero out the backup midnight reset variable
}

//Calculates each of the variables that wunderground is expecting
void calcWeather()
{
    //current winddir, current windspeed, windgustmph, and windgustdir are calculated
    every 100ms throughout the day

    //Calc windspdmpg_avg2m
    float temp = 0;
    for(int i = 0 ; i < 120 ; i++)
        temp += windspdavg[i];
    temp /= 120.0;
    windspdmpg_avg2m = temp;

    //Calc winddir_avg2m, Wind Direction
    //You can't just take the average. Google "mean of circular quantities" for more
    info
    //We will use the Mitsuta method because it doesn't require trig functions
    //And because it sounds cool.
    //Based on: http://abelian.org/vlf/bearings.html
    //Based on: http://stackoverflow.com/questions/1813483/averaging-angles-again
    long sum = winddiravg[0];
    int D = winddiravg[0];
    for(int i = 1 ; i < WIND_DIR_AVG_SIZE ; i++)
    {
        int delta = winddiravg[i] - D;

        if(delta < -180)
            D += delta + 360;
        else if(delta > 180)
            D += delta - 360;
        else

```

```
D += delta;

sum += D;
}
winddir_avg2m = sum / WIND_DIR_AVG_SIZE;
if(winddir_avg2m >= 360) winddir_avg2m -= 360;
if(winddir_avg2m < 0) winddir_avg2m += 360;

//Calc windgustmph_10m
//Calc windgustdir_10m
//Find the largest windgust in the last 10 minutes
windgustmph_10m = 0;
windgustdir_10m = 0;
//Step through the 10 minutes
for(int i = 0; i < 10 ; i++)
{
    if(windgust_10m[i] > windgustmph_10m)
    {
        windgustmph_10m = windgust_10m[i];
        windgustdir_10m = windgustdirection_10m[i];
    }
}

//Calc humidity
humidity = myHumidity.readHumidity();
//float temp_h = myHumidity.readTemperature();
//Serial.print(" TempH:");
//Serial.print(temp_h, 2);

//Calc tempf from pressure sensor
tempf = myPressure.readTempF();
//Serial.print(" TempP:");
//Serial.print(tempf, 2);

//Total rainfall for the day is calculated within the interrupt
//Calculate amount of rainfall for the last 60 minutes
rainin = 0;
for(int i = 0 ; i < 60 ; i++)
    rainin += rainHour[i];
```

```

    //Calc pressure
    pressure = myPressure.readPressure();

    //Calc dewptf

    //Calc light level
    light_lvl = get_light_level();

    //Calc battery level
    batt_lvl = get_battery_level();

    //Lightning is checked in the main loop
}

//Returns the voltage of the light sensor based on the 3.3V rail
//This allows us to ignore what VCC might be (an Arduino plugged into USB has VCC
of 4.5 to 5.2V)
float get_light_level()
{
    float operatingVoltage = averageAnalogRead(REFERENCE_3V3);

    float lightSensor = averageAnalogRead(LIGHT);

    operatingVoltage = 3.3 / operatingVoltage; //The reference voltage is 3.3V

    lightSensor *= operatingVoltage;

    return(lightSensor);
}

//Returns the voltage of the raw pin based on the 3.3V rail
//The battery can ranges from 4.2V down to around 3.3V
//This function allows us to ignore what VCC might be (an Arduino plugged into USB
has VCC of 4.5 to 5.2V)
//The weather shield has a pin called RAW (VIN) fed through through two 5%
resistors and connected to A2 (BATT):
//3.9K on the high side (R1), and 1K on the low side (R2)
float get_battery_level()
{
    float operatingVoltage = averageAnalogRead(REFERENCE_3V3);

    float rawVoltage = averageAnalogRead(BATT);

```

```
operatingVoltage = 3.30 / operatingVoltage; //The reference voltage is 3.3V

rawVoltage *= operatingVoltage; //Convert the 0 to 1023 int to actual voltage on
BATT pin

rawVoltage *= 4.90; //(3.9k+1k)/1k - multiply BATT voltage by the voltage divider
to get actual system voltage

return(rawVoltage);
}

//Returns the instataneous wind speed
float get_wind_speed()
{
    float deltaTime = millis() - lastWindCheck; //750ms

    deltaTime /= 1000.0; //Covert to seconds

    float windSpeed = (float)windClicks / deltaTime; //3 / 0.750s = 4

    windClicks = 0; //Reset and start watching for new wind
    lastWindCheck = millis();

    windSpeed *= 1.492; //4 * 1.492 = 5.968MPH

    /* Serial.println();
    Serial.print("Windspeed:");
    Serial.println(windSpeed);*/

    return(windSpeed);
}

int get_wind_direction()
// read the wind direction sensor, return heading in degrees
{
    unsigned int adc;

    adc = averageAnalogRead(WDIR); // get the current reading from the sensor
```

// The following table is ADC readings for the wind direction sensor output, sorted from low to high.

// Each threshold is the midpoint between adjacent headings. The output is degrees for that ADC reading.

// Note that these are not in compass degree order! See Weather Meters datasheet for more information.

```
    if (adc < 380) return (113);
    if (adc < 393) return (68);
    if (adc < 414) return (90);
    if (adc < 456) return (158);
    if (adc < 508) return (135);
    if (adc < 551) return (203);
    if (adc < 615) return (180);
    if (adc < 680) return (23);
    if (adc < 746) return (45);
    if (adc < 801) return (248);
    if (adc < 833) return (225);
    if (adc < 878) return (338);
    if (adc < 913) return (0);
    if (adc < 940) return (293);
    if (adc < 967) return (315);
    if (adc < 990) return (270);
    return (-1); // error, disconnected?
}

//Reports the weather string to the Imp
void reportWeather()
{
    calcWeather(); //Go calc all the various sensors

    Serial.print("$,winddir=");
    Serial.print(winddir);
    Serial.print(",windspeedmph=");
    Serial.print(windspeedmph, 1);
    Serial.print(",windgustmph=");
    Serial.print(windgustmph, 1);
    Serial.print(",windgustdir=");
    Serial.print(windgustdir);
    Serial.print(",windspdmph_avg2m=");
    Serial.print(windspdmph_avg2m, 1);
    Serial.print(",winddir_avg2m=");
    Serial.print(winddir_avg2m);
```

```
Serial.print(",windgustmph_10m=");
Serial.print(windgustmph_10m, 1);
Serial.print(",windgustdir_10m=");
Serial.print(windgustdir_10m);
Serial.print(",humidity=");
Serial.print(humidity, 1);
Serial.print(",tempf=");
Serial.print(tempf, 1);
Serial.print(",rainin=");
Serial.print(rainin, 2);
Serial.print(",dailyrainin=");
Serial.print(dailyrainin, 2);

Serial.print(","); //Don't print pressure= because the agent will be doing calcs
on the number
Serial.print(pressure, 2);
Serial.print(",batt_lvl=");
Serial.print(batt_lvl, 2);
Serial.print(",light_lvl=");
Serial.print(light_lvl, 2);

#ifdef LIGHTNING_ENABLED
    Serial.print(",lightning_distance=");
    Serial.print(lightning_distance);
#endif

Serial.print(",");
Serial.println("#,");

//Test string
//Serial.println("$,winddir=270,windspeedmph=0.0,windgustmph=0.0,windgustdir=0,wi
ndspdmph_avg2m=0.0,winddir_avg2m=12,windgustmph_10m=0.0,windgustdir_10m=0,humidity=998.
0,tempf=-1766.2,rainin=0.00,dailyrainin=0.00,-
999.00,batt_lvl=16.11,light_lvl=3.32,#,");
}

//Takes an average of readings on a given pin
//Returns the average
int averageAnalogRead(int pinToRead)
{
    byte numberOfReadings = 8;
    unsigned int runningValue = 0;
```

```
    for(int x = 0 ; x < numberOfReadings ; x++)
        runningValue += analogRead(pinToRead);
    runningValue /= numberOfReadings;

    return(runningValue);
}

//The following is for the AS3935 lightning sensor
#ifdef ENABLE_LIGHTNING
byte readLightning(void)
{
    byte distance = 0;

    //Check to see if we have lightning!
    if(digitalRead(LIGHTNING_IRQ) == HIGH)
    {
        // first step is to find out what caused interrupt
        // as soon as we read interrupt cause register, irq pin goes low
        int irqSource = AS3935.interruptSource();

        // returned value is bitmap field, bit 0 - noise level too high, bit 2 -
disturber detected, and finally bit 3 - lightning!
        if (irqSource & 0b0001)
        {
            //Serial.println("Noise level too high, try adjusting noise floor");
        }

        if (irqSource & 0b0100)
        {
            //Serial.println("Disturber detected");
            distance = 64;
        }

        if (irqSource & 0b1000)
        {
            // need to find how far that lightning stroke, function returns
approximate distance in kilometers,
            // where value 1 represents storm in detector's near vicinity, and
63 - very distant, out of range stroke
            // everything in between is just distance in kilometers
            distance = AS3935.lightningDistanceKm();

            //Serial.print("Lightning: ");

```

```
//Serial.print(lightning_distance, DEC);
//Serial.println(" km");

//The AS3935 remembers the nearest strike distance. For example 15km
away then 10, then overhead all following
//distances (10, 20, 30) will instead output as 'Storm overhead,
watch out!'. Resetting the chip erases this.
    lightning_init();
}
}

return(distance);
}

void startLightning(void)
{
    pinMode(slaveSelectPin, OUTPUT); // set the slaveSelectPin as an output:

    pinMode(LIGHTNING_IRQ, INPUT_PULLUP); //Set IRQ pin as input

    SPI.begin(); //Start SPI

    SPI.setDataMode(SPI_MODE1); // NB! chip uses SPI MODE1

    SPI.setClockDivider(SPI_CLOCK_DIV16); //Uno 16MHz / 16 = 1MHz

    SPI.setBitOrder(MSBFIRST); // and chip is MSB first

    lightning_init(); //Setup the values for the sensor

    Serial.println("Lightning sensor online");
}

void lightning_init()
{
    AS3935.reset(); // reset all internal register values to defaults

    // if lightning detector can not tune tank circuit to required tolerance,
    // calibration function will return false
    if(!AS3935.calibrate())
    {
```

```
        Serial.println("Tuning out of range, check your wiring, your sensor and
make sure physics laws have not changed!");
    }

    AS3935.setOutdoors(); //The weather station is outdoors

    AS3935.enableDisturbers(); //We want to know if a man-made event happens
    AS3935.setNoiseFloor(3); //See table 16 of the AS3935 datasheet. 4-6 works. This
was found through experimentation.

    //printAS3935Registers();
}

/*void printAS3935Registers()
{
    int noiseFloor = AS3935.getNoiseFloor();
    int spikeRejection = AS3935.getSpikeRejection();
    int watchdogThreshold = AS3935.getWatchdogThreshold();
    Serial.print("Noise floor is: ");
    Serial.println(noiseFloor, DEC);
    Serial.print("Spike rejection is: ");
    Serial.println(spikeRejection, DEC);
    Serial.print("Watchdog threshold is: ");
    Serial.println(watchdogThreshold, DEC);
}*/

byte SPITransfer(byte sendByte)
{
    return SPI.transfer(sendByte);
}

#endif
```

ANEXO VI: CÓDIGO DEVICE IMP IDE

```
// This agent gathers data from the device and pushes to Wunderground
// Talks to wunderground rapid fire server (updates of up to once every 10 sec)
// by: Nathan Seidle
//      SparkFun Electronics
// date: October 4, 2013
// license: BeerWare
//      Please use, reuse, and modify this code as you need.
//      We hope it saves you some time, or helps you learn something!
//      If you find it handy, and we meet some day, you can buy me a beer or iced t
ea in return.

// Example incoming serial string from device:
// $,winddir=270,windspeedmph=0.0,windgustmph=0.0,windgustdir=0,windspeedmph_avg2m=0.0,wi
nddir_avg2m=12,windgustmph_10m=0.0,windgustdir_10m=0,humidity=998.0,tempf=-1766.2,raini
n=0.00,dailyrainin=0.00,pressure=-999.00,batt_lvl=16.11,light_lvl=3.32,#

local STATION_ID = "KCOBOULD95";
local STATION_PW = "password"; //Note that you must only use alphanumeric in your pass
word. Http post won't work otherwise.

local sparkfun_publicKey = "dZ4EVmE8yGCRGx5XRX1W";
local sparkfun_privateKey = "privatekey";

local LOCAL_ALTITUDE_METERS = 1638; //Accurate for the roof on my house

local midnightReset = false; //Keeps track of a once per day cumulative rain reset

local local_hour_offset = 7; //Mountain time is 7 hours off GMT

const MAX_PROGRAM_SIZE = 0x20000;
const ARDUINO_BLOB_SIZE = 128;
program <- null;

//-----
//-----

html <- @"<HTML>
<BODY>

<form method='POST' enctype='multipart/form-data'>
Program the ATmega328 via the Imp.<br/><br/>
Step 1: Select an Intel HEX file to upload: <input type=file name=hexfile><br/>
Step 2: <input type=submit value=Press> to upload the file.<br/>
Step 3: Check out your Arduino<br/>
</form>

</BODY>
```

```
</HTML>
```

```
";
```

```
//-----  
-----
```

```
// Parses a HTTP POST in multipart/form-data format
```

```
function parse_hexpost(req, res) {  
  local boundary = req.headers["content-type"].slice(30);  
  local bindex = req.body.find(boundary);  
  local hstart = bindex + boundary.len();  
  local bstart = req.body.find("\r\n\r\n", hstart) + 4;  
  local fstart = req.body.find("\r\n\r\n--" + boundary + "--", bstart);  
  return req.body.slice(bstart, fstart);  
}
```

```
//-----  
-----
```

```
// Parses a hex string and turns it into an integer
```

```
function hextoint(str) {  
  local hex = 0x0000;  
  foreach (ch in str) {  
    local nibble;  
    if (ch >= '0' && ch <= '9') {  
      nibble = (ch - '0');  
    } else {  
      nibble = (ch - 'A' + 10);  
    }  
    hex = (hex << 4) + nibble;  
  }  
  return hex;  
}
```

```
//-----  
-----
```

```
// Breaks the program into chunks and sends it to the device
```

```
function send_program() {  
  if (program != null && program.len() > 0) {  
    local addr = 0;  
    local pline = {};  
    local max_addr = program.len();  
  
    device.send("burn", {first=true});  
    while (addr < max_addr) {  
      program.seek(addr);  
      pline.data <- program.readblob(ARDUINO_BLOB_SIZE);  
      pline.addr <- addr / 2; // Address space is 16-bit  
      device.send("burn", pline)  
      addr += pline.data.len();  
    }  
  }  
}
```

```

    }
    device.send("burn", {last=true});
}
}

//-----
// Parse the hex into an array of blobs
function parse_hexfile(hex) {

    try {
        // Look at this doc to work out what we need and don't. Max is about 122kb.
        // https://bluegiga.zendesk.com/entries/42713448--REFERENCE-Updating-BLE11x-fir
        mware-using-UART-DFU
        server.log("Parsing hex file");

        // Create and blank the program blob
        program = blob(0x20000); // 128k maximum
        for (local i = 0; i < program.len(); i++) program.writen(0x00, 'b');
        program.seek(0);

        local maxaddress = 0, from = 0, to = 0, line = "", offset = 0x00000000;
        do {
            if (to < 0 || to == null || to >= hex.len()) break;
            from = hex.find(":", to);

            if (from < 0 || from == null || from+1 >= hex.len()) break;
            to = hex.find(":", from+1);

            if (to < 0 || to == null || from >= to || to >= hex.len()) break;
            line = hex.slice(from+1, to);
            // server.log(format("[%d,%d] => %s", from, to, line));

            if (line.len() > 10) {
                local len = hextoint(line.slice(0, 2));
                local addr = hextoint(line.slice(2, 6));
                local type = hextoint(line.slice(6, 8));

                // Ignore all record types except 00, which is a data record.
                // Look out for 02 records which set the high order byte of the address
                space

                if (type == 0) {
                    // Normal data record
                } else if (type == 4 && len == 2 && addr == 0 && line.len() > 12) {
                    // Set the offset
                    offset = hextoint(line.slice(8, 12)) << 16;
                    if (offset != 0) {
                        server.log(format("Set offset to 0x%08X", offset));
                    }
                }
            }
        } while (true);
    }
}

```

```
        continue;
    } else {
        server.log("Skipped: " + line)
        continue;
    }

    // Read the data from 8 to the end (less the last checksum byte)
    program.seek(offset + addr)
    for (local i = 8; i < 8+(len*2); i+=2) {
        local datum = hextoint(line.slice(i, i+2));
        program.written(datum, 'b')
    }

    // Checking the checksum would be a good idea but skipped for now
    local checksum = hextoint(line.slice(-2));

    /// Shift the end point forward
    if (program.tell() > maxaddress) maxaddress = program.tell();

    }
} while (from != null && to != null && from < to);

// Crop, save and send the program
server.log(format("Max address: 0x%08x", maxaddress));
program.resize(maxaddress);
send_program();
server.log("Free RAM: " + (imp.getmemoryfree()/1024) + " kb")
return true;

} catch (e) {
    server.log(e)
    return false;
}

}

//-----
// Handle the agent requests
http.onrequest(function (req, res) {
    // return res.send(400, "Bad request");
    // server.log(req.method + " to " + req.path)
    if (req.method == "GET") {
        res.send(200, html);
    } else if (req.method == "POST") {

        if ("content-type" in req.headers) {
            if (req.headers["content-type"].len() >= 19
                && req.headers["content-type"].slice(0, 19) == "multipart/form-data") {
```

```

    local hex = parse_hexpost(req, res);
    if (hex == "") {
        res.header("Location", http.agenturl());
        res.send(302, "HEX file uploaded");
    } else {
        device.on("done", function(ready) {
            res.header("Location", http.agenturl());
            res.send(302, "HEX file uploaded");
            server.log("Programming completed")
        })
        server.log("Programming started")
        parse_hexfile(hex);
    }
} else if (req.headers["content-type"] == "application/json") {
    local json = null;
    try {
        json = http.jsondecode(req.body);
    } catch (e) {
        server.log("JSON decoding failed for: " + req.body);
        return res.send(400, "Invalid JSON data");
    }
    local log = "";
    foreach (k,v in json) {
        if (typeof v == "array" || typeof v == "table") {
            foreach (k1,v1 in v) {
                log += format("%s[%s] => %s, ", k, k1, v1.tostring());
            }
        } else {
            log += format("%s => %s, ", k, v.tostring());
        }
    }
    server.log(log)
    return res.send(200, "OK");
} else {
    return res.send(400, "Bad request");
}
} else {
    return res.send(400, "Bad request");
}
}
})

//-----
//-----

// Handle the device coming online
device.on("ready", function(ready) {
    if (ready) send_program();
});

```

```
//-----  
-----  
  
// When we hear something from the device, split it apart and post it  
device.on("postToInternet", function(dataString) {  
  
    //server.log("Incoming: " + dataString);  
  
    //Break the incoming string into pieces by comma  
    a <- mysplit(dataString, ',');  
  
    if(a[0] != "$" || a[16] != "#")  
    {  
        server.log(format("Error: incorrect frame received (%s, %s)", a[0], a[16]));  
        server.log(format("Received: %s", dataString));  
        return(0);  
    }  
  
    //Pull the various bits from the blob  
  
    //a[0] is $  
    local winddir = a[1];  
    local windspeedmph = a[2];  
    local windgustmph = a[3];  
    local windgustdir = a[4];  
    local windspdmpg_avg2m = a[5];  
    local winddir_avg2m = a[6];  
    local windgustmph_10m = a[7];  
    local windgustdir_10m = a[8];  
    local humidity = a[9];  
    local tempf = a[10];  
    local rainin = a[11];  
    local dailyrainin = a[12];  
    local pressure = a[13].toFloat();  
    local batt_lvl = a[14];  
    local light_lvl = a[15];  
    //a[16] is #  
  
    server.log(tempf);  
  
    //Correct for the actual orientation of the weather station  
    //For my station the north indicator is pointing due west  
    winddir = windCorrect(winddir);  
    windgustdir = windCorrect(windgustdir);  
    winddir_avg2m = windCorrect(winddir_avg2m);  
    windgustdir_10m = windCorrect(windgustdir_10m);  
  
    //Correct for negative temperatures. This is fixed in the latest libraries: https://
```

```
/learn.sparkfun.com/tutorials/mpl3115a2-pressure-sensor-hookup-guide
currentTemp <- mysplit(tempf, '=');
local badTempf = currentTemp[1].tointeger();
if(badTempf > 200)
{
    local tempc = (badTempf - 32) * 5/9; //Convert F to C
    tempc = (tempc<<24)>>24; //Force this 8 bit value into 32 bit variable
    tempc = ~(tempc) + 1; //Take 2s compliment
    tempc *= -1; //Assign negative sign
    tempf = tempc * 9/5 + 32; //Convert back to F
    tempf = "tempf=" + tempf; //put a string on it
}

//Correct for humidity out of bounds
currentHumidity <- mysplit(humidity, '=');
if(currentHumidity[1].tointeger() > 99) humidity = "humidity=99";
if(currentHumidity[1].tointeger() < 0) humidity = "humidity=0";

//Turn Pascal pressure into baromin (Inches Mercury at Altimeter Setting)
local baromin = "baromin=" + convertToInHg(pressure);

//Calculate a dew point
currentHumidity <- mysplit(humidity, '=');
currentTempF <- mysplit(tempf, '=');
local dewptf = "dewptf=" + calcDewPoint(currentHumidity[1].tointeger(), currentTemp
F[1].tointeger());

//Now we form the large string to pass to wunderground
local strMainSite = "http://rtupdate.wunderground.com/weatherstation/updateweathers
tation.php";

local strID = "ID=" + STATION_ID;
local strPW = "PASSWORD=" + STATION_PW;

//Form the current date/time
//Note: .month is 0 to 11!
local currentTime = date(time(), 'u');
local strCT = "dateutc=";
strCT += currentTime.year + "-" + format("%02d", currentTime.month + 1) + "-" + for
mat("%02d", currentTime.day);
strCT += "+" + format("%02d", currentTime.hour) + "%3A" + format("%02d", currentTim
e.min) + "%3A" + format("%02d", currentTime.sec);
//Not sure if wunderground expects the + or a %2B. We shall see.
//server.log(strCT);

local bigString = strMainSite;
bigString += "?" + strID;
bigString += "&" + strPW;
bigString += "&" + strCT;
```

```
bigString += "&" + winddir;
bigString += "&" + windspeedmph;
bigString += "&" + windgustmph;
bigString += "&" + windgustdir;
bigString += "&" + windspdmp_h_avg2m;
bigString += "&" + winddir_avg2m;
bigString += "&" + windgustmph_10m;
bigString += "&" + windgustdir_10m;
bigString += "&" + humidity;
bigString += "&" + tempf;
bigString += "&" + rainin;
bigString += "&" + dailyrainin;
bigString += "&" + baromin;
bigString += "&" + dewptf;
//bigString += "&" + weather;
//bigString += "&" + clouds;
bigString += "&" + "softwaretype=SparkFunWeatherImp"; //Cause we can
bigString += "&" + "realtime=1"; //You better believe it!
bigString += "&" + "rttfreq=10"; //Set rapid fire freq to once every 10 seconds
bigString += "&" + "action=updateraw";

//server.log("string to send: " + bigString);

//Push to Wunderground
local request = http.post(bigString, {}, "");
local response = request.sendsync();
server.log("Wunderground response = " + response.body);
server.log(batt_lvl + " " + light_lvl);

//Get the local time that this measurement was taken
local localMeasurementTime = "measurementtime=" + calcLocalTime();

//Now post to data.sparkfun.com
//Here is a list of datums: measurementTime, winddir, windspeedmph, windgustmph, wi
ndgustdir, windspdmp_h_avg2m, winddir_avg2m, windgustmph_10m, windgustdir_10m, humidity,
tempf, rainin, dailyrainin, baromin, dewptf, batt_lvl, light_lvl

//Now we form the large string to pass to sparkfun
local strSparkFun = "http://data.sparkfun.com/input/";
local privateKey = "private_key=" + sparkfun_privateKey;

bigString = strSparkFun;
bigString += sparkfun_publicKey;
bigString += "?" + privateKey;
bigString += "&" + localMeasurementTime;
bigString += "&" + winddir;
bigString += "&" + windspeedmph;
bigString += "&" + windgustmph;
bigString += "&" + windgustdir;
bigString += "&" + windspdmp_h_avg2m;
```

```
bigString += "&" + winddir_avg2m;
bigString += "&" + windgustmph_10m;
bigString += "&" + windgustdir_10m;
bigString += "&" + humidity;
bigString += "&" + tempf;
bigString += "&" + rainin;
bigString += "&" + dailyrainin;
bigString += "&" + baromin;
bigString += "&" + dewptf;
bigString += "&" + batt_lvl;
bigString += "&" + light_lvl;

//Push to SparkFun
local request = http.get(bigString);
local response = request.sendsync();
server.log("SparkFun response = " + response.body);

//Check to see if we need to send a midnight reset
checkMidnight(1);

server.log("Update complete!");
});

//Given a string, break out the direction, correct by some value
//Return a string
function windCorrect(direction) {
  temp <- mysplit(direction, '=');

  //My station's North arrow is pointing due west
  //So correct by 90 degrees
  local dir = temp[1].tointeger() - 90;
  if(dir < 0) dir += 360;
  return(temp[0] + "=" + dir);
}

//With relative humidity and temp, calculate a dew point
//From: http://ag.arizona.edu/azmet/dewpoint.html
function calcDewPoint(relativeHumidity, tempF) {
  local tempC = (tempF - 32) * 5 / 9.0;

  local L = math.log(relativeHumidity / 100.0);
  local M = 17.27 * tempC;
  local N = 237.3 + tempC;
  local B = (L + (M / N)) / 17.27;
  local dewPoint = (237.3 * B) / (1.0 - B);

  //Result is in C
  //Convert back to F
  dewPoint = dewPoint * 9 / 5.0 + 32;
```

```
//server.log("rh=" + relativeHumidity + " tempF=" + tempF + " tempC=" + tempC);
//server.log("DewPoint = " + dewPoint);
return(dewPoint);
}

function checkMidnight(ignore) {
    //Check to see if it's midnight. If it is, send @ to Arduino to reset time based va
riables

    //Get the local time that this measurement was taken
    local localTime = calcLocalTime();

    //server.log("Local hour = " + format("%c", localTime[0]) + format("%c", localTime[
1]));

    if(localTime[0].tochar() == "0" && localTime[1].tochar() == "4")
    {
        if(midnightReset == false)
        {
            server.log("Sending midnight reset");
            midnightReset = true; //We should only reset once
            device.send("sendMidnightReset", 1);
        }
    }
    else {
        midnightReset = false; //Reset our state
    }
}

//Recording to a google doc is a bit tricky. Many people have found ways of posting
//to a google form to get data into a spreadsheet. This requires a https connection
//so we use pushingbox to handle the secure connection.
//See http://productforums.google.com/forum/#!topic/docs/f4hJKF1OQOw for more info
//To push two items I had to use a GET instead of a post
function recordLevels(batt, light) {

    //Smash it all together
    local stringToSend = "http://api.pushingbox.com/pushingbox?devid=vB0A3446EBB4828F";
    stringToSend = stringToSend + "&" + batt;
    stringToSend = stringToSend + "&" + light;
    //server.log("string to send: " + stringToSend); //Debugging

    //Push to internet
    local request = http.post(stringToSend, {}, "");
    local response = request.sendsync();
    //server.log("Google response=" + response.body);

    server.log("Post to spreadsheet complete.")
}
```

```
//Given pressure in pascals, convert the pressure to Altimeter Setting, inches mercury
function convertToInHg(pressure_Pa)
{
    local pressure_mb = pressure_Pa / 100; //pressure is now in millibars, 1 pascal = 0
    .01 millibars

    local part1 = pressure_mb - 0.3; //Part 1 of formula
    local part2 = 8.42288 / 100000.0;
    local part3 = math.pow((pressure_mb - 0.3), 0.190284);
    local part4 = LOCAL_ALTITUDE_METERS / part3;
    local part5 = (1.0 + (part2 * part4));
    local part6 = math.pow(part5, (1.0/0.190284));
    local altimeter_setting_pressure_mb = part1 * part6; //Output is now in adjusted mi
    llibars
    local baromin = altimeter_setting_pressure_mb * 0.02953;
    //server.log(format("%s", baromin));
    return(baromin);
}

//From Hugo: http://forums.electricimp.com/discussion/915/processing-nmea-0183-gps-strings/p1
//You rock! Thanks Hugo!
function mysplit(a, b) {
    local ret = [];
    local field = "";
    foreach(c in a) {
        if (c == b) {
            // found separator, push field
            ret.push(field);
            field="";
        } else {
            field += c.tochar(); // append to field
        }
    }
    // Push the last field
    ret.push(field);
    return ret;
}

//Given UTC time and a local offset and a date, calculate the local time
//Includes a daylight savings time calc for the US
function calcLocalTime()
{
    //Get the time that this measurement was taken
    local currentTime = date(time(), 'u');
    local hour = currentTime.hour; //Most of the work will be on the current hour

    //Since 2007 DST starts on the second Sunday in March and ends the first Sunday of
```

November

```
//Let's just assume it's going to be this way for awhile (silly US government!)
//Example from: http://stackoverflow.com/questions/5590429/calculating-daylight-savings-time-from-only-date

//The Imp .month returns 0-11. DoW expects 1-12 so we add one.
local month = currentTime.month + 1;

local DoW = day_of_week(currentTime.year, month, currentTime.day); //Get the day of
the week. 0 = Sunday, 6 = Saturday
local previousSunday = currentTime.day - DoW;

local dst = false; //Assume we're not in DST
if(month > 3 && month < 11) dst = true; //DST is happening!

//In March, we are DST if our previous Sunday was on or after the 8th.
if (month == 3)
{
    if(previousSunday >= 8) dst = true;
}
//In November we must be before the first Sunday to be dst.
//That means the previous Sunday must be before the 1st.
if(month == 11)
{
    if(previousSunday <= 0) dst = true;
}

if(dst == true)
{
    hour++; //If we're in DST add an extra hour
}

//Convert UTC hours to local current time using local_hour
if(hour < local_hour_offset)
    hour += 24; //Add 24 hours before subtracting local offset
hour -= local_hour_offset;

local AMPM = "AM";
if(hour > 12)
{
    hour -= 12; //Get rid of military time
    AMPM = "PM";
}
if(hour == 0) hour = 12; //Midnight edge case

currentTime = format("%02d", hour) + "%3A" + format("%02d", currentTime.min) + "%3A"
+ format("%02d", currentTime.sec) + "%20" + AMPM;
//server.log("Local time: " + currentTime);
return(currentTime);
}
```

```
//Given the current year/month/day
//Returns 0 (Sunday) through 6 (Saturday) for the day of the week
//Assumes we are operating in the 2000-2099 century
//From: http://en.wikipedia.org/wiki/Calculating\_the\_day\_of\_the\_week
function day_of_week(year, month, day)
{

    //offset = centuries table + year digits + year fractional + month lookup + date
    local centuries_table = 6; //We assume this code will only be used from year 2000 to
year 2099
    local year_digits;
    local year_fractional;
    local month_lookup;
    local offset;

    //Example Feb 9th, 2011

    //First boil down year, example year = 2011
    year_digits = year % 100; //year_digits = 11
    year_fractional = year_digits / 4; //year_fractional = 2

    switch(month) {
case 1:
    month_lookup = 0; //January = 0
    break;
case 2:
    month_lookup = 3; //February = 3
    break;
case 3:
    month_lookup = 3; //March = 3
    break;
case 4:
    month_lookup = 6; //April = 6
    break;
case 5:
    month_lookup = 1; //May = 1
    break;
case 6:
    month_lookup = 4; //June = 4
    break;
case 7:
    month_lookup = 6; //July = 6
    break;
case 8:
    month_lookup = 2; //August = 2
    break;
case 9:
    month_lookup = 5; //September = 5
```

```
    break;
case 10:
    month_lookup = 0; //October = 0
    break;
case 11:
    month_lookup = 3; //November = 3
    break;
case 12:
    month_lookup = 5; //December = 5
    break;
default:
    month_lookup = 0; //Error!
    return(-1);
}

offset = centuries_table + year_digits + year_fractional + month_lookup + day;
//offset = 6 + 11 + 2 + 3 + 9 = 31
offset %= 7; // 31 % 7 = 3 Wednesday!

return(offset); //Day of week, 0 to 6

//Example: May 11th, 2012
//6 + 12 + 3 + 1 + 11 = 33
//5 = Friday! It works!

//Devised by Tomohiko Sakamoto in 1993, it is accurate for any Gregorian date:
/*t <- [ 0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4];
if(month < 3) year--;
//year = month < 3;
return (year + year/4 - year/100 + year/400 + t[month-1] + day) % 7;
//return 4;
*/
}
```

ANEXO VII: CÓDIGO AGENT

```
// Reads data from a station and pushes it to an agent
// Agent then pushes the weather data to Wunderground
// by: Nathan Seidle
//      SparkFun Electronics
// date: October 4, 2013
// license: BeerWare
//      Please use, reuse, and modify this code as you need.
//      We hope it saves you some time, or helps you learn something!
//      If you find it handy, and we meet some day, you can buy me a beer or iced t
ea in return.

local rxLEDToggle = 1; // These variables keep track of rx/tx LED toggling status
local txLEDToggle = 1;

local NOCHAR = -1;

//-----
//Everything below is used for bootloading

server.log("Device started, impee_id " + hardware.getimpeeid() + " and mac = " + imp.ge
tmacaddress() );

//-----
-----
// Uart57 for TX/RX
SERIAL <- hardware.uart57;
SERIAL.configure(115200, 8, PARITY_NONE, 1, NO_CTSRTS);

// Set pin1 high for normal operation
// Set pin1 low to reset a standard Arduino
RESET <- hardware.pin1;
//RESET.configure(DIGITAL_OUT); //This causes the board to stick in reset state? Not su
re.
RESET.write(1); //Leave Arduino in normal (non-reset) state

// Pin 9 is the yellow LED on the Imp Shield
ACTIVITY <- hardware.pin9;
ACTIVITY.configure(DIGITAL_OUT);
ACTIVITY.write(1);

// Pin 8 is the orange LED
LINK <- hardware.pin8;
LINK.configure(DIGITAL_OUT);
LINK.write(1);
```

```
// Sequence number
__seq <- 0x28;

//-----

/* STK500 constants list, from AVRDUDE */
const MESSAGE_START      = 0x1B;
const TOKEN               = 0x0E;
const STK_OK              = 0x10;
const STK_FAILED          = 0x11; // Not used
const STK_UNKNOWN         = 0x12; // Not used
const STK_NODEVICE        = 0x13; // Not used
const STK_INSYNC          = 0x14; // ' '
const STK_NOSYNC          = 0x15; // Not used
const ADC_CHANNEL_ERROR   = 0x16; // Not used
const ADC_MEASURE_OK      = 0x17; // Not used
const PWM_CHANNEL_ERROR   = 0x18; // Not used
const PWM_ADJUST_OK       = 0x19; // Not used
const CRC_EOP             = 0x20; // 'SPACE'
const STK_GET_SYNC        = 0x30; // '0'
const STK_GET_SIGN_ON     = 0x31; // '1'
const STK_SET_PARAMETER   = 0x40; // '@'
const STK_GET_PARAMETER   = 0x41; // 'A'
const STK_SET_DEVICE      = 0x42; // 'B'
const STK_SET_DEVICE_EXT  = 0x45; // 'E'
const STK_ENTER_PROGMODE  = 0x50; // 'P'
const STK_LEAVE_PROGMODE  = 0x51; // 'Q'
const STK_CHIP_ERASE      = 0x52; // 'R'
const STK_CHECK_AUTOINC   = 0x53; // 'S'
const STK_LOAD_ADDRESS    = 0x55; // 'U'
const STK_UNIVERSAL       = 0x56; // 'V'
const STK_PROG_FLASH      = 0x60; // '`'
const STK_PROG_DATA       = 0x61; // 'a'
const STK_PROG_FUSE       = 0x62; // 'b'
const STK_PROG_LOCK       = 0x63; // 'c'
const STK_PROG_PAGE       = 0x64; // 'd'
const STK_PROG_FUSE_EXT   = 0x65; // 'e'
const STK_READ_FLASH      = 0x70; // 'p'
const STK_READ_DATA       = 0x71; // 'q'
const STK_READ_FUSE       = 0x72; // 'r'
const STK_READ_LOCK       = 0x73; // 's'
const STK_READ_PAGE       = 0x74; // 't'
const STK_READ_SIGN       = 0x75; // 'u'
const STK_READ_OSCCAL     = 0x76; // 'v'
const STK_READ_FUSE_EXT   = 0x77; // 'w'
const STK_READ_OSCCAL_EXT = 0x78; // 'x'

//-----
```

```

-----
function HEXDUMP(buf, len = null) {
  if (buf == null) return "null";
  if (len == null) {
    len = (typeof buf == "blob") ? buf.tell() : buf.len();
  }

  local dbg = "";
  for (local i = 0; i < len; i++) {
    local ch = buf[i];
    dbg += format("0x%02X ", ch);
  }

  return format("%s (%d bytes)", dbg, len)
}

```

```

//-----
function SERIAL_READ(len = 100, timeout = 300) {

  local rxbuf = blob(len);
  local write = rxbuf.written.bindenv(rxbuf);
  local read = SERIAL.read.bindenv(SERIAL);
  local hw = hardware;
  local ms = hw.millis.bindenv(hw);
  local started = ms();

  local charsRead = 0;
  LINK.write(0); //Turn LED on
  do {
    local ch = read();
    if (ch != -1) {
      write(ch, 'b')
      charsRead++;
      if(charsRead == len) break;
    }
  } while (ms() - started < timeout);
  LINK.write(1); //Turn LED off

  // Clean up any extra bytes
  while (SERIAL.read() != -1);

  if (rxbuf.tell() == 0) {
    return null;
  } else {
    return rxbuf;
  }
}

```

```
//-----  
-----  
function execute(command = null, param = null, response_length = 100, response_timeout  
= 300) {  
  
    local send_buffer = null;  
    if (command == null) {  
        send_buffer = format("%c", CRC_EOP);  
    } else if (param == null) {  
        send_buffer = format("%c%c", command, CRC_EOP);  
    } else if (typeof param == "array") {  
        send_buffer = format("%c", command);  
        foreach (datum in param) {  
            switch (typeof datum) {  
                case "string":  
                case "blob":  
                case "array":  
                case "table":  
                    foreach (adat in datum) {  
                        send_buffer += format("%c", adat);  
                    }  
                    break;  
                default:  
                    send_buffer += format("%c", datum);  
            }  
        }  
        send_buffer += format("%c", CRC_EOP);  
    } else {  
        send_buffer = format("%c%c%c", command, param, CRC_EOP);  
    }  
  
    // server.log("Sending: " + HEXDUMP(send_buffer));  
    SERIAL.write(send_buffer);  
  
    local resp_buffer = SERIAL_READ(response_length+2, response_timeout);  
    // server.log("Received: " + HEXDUMP(resp_buffer));  
  
    assert(resp_buffer != null);  
    assert(resp_buffer.tell() >= 2);  
    assert(resp_buffer[0] == STK_INSYNC);  
    assert(resp_buffer[resp_buffer.tell()-1] == STK_OK);  
  
    local tell = resp_buffer.tell();  
    if (tell == 2) return blob(0);  
    resp_buffer.seek(1);  
    return resp_buffer.readblob(tell-2);  
}
```

```

//-----
function check_duino() {
    // Clear the read buffer
    SERIAL_READ();

    // Check everything we can check to ensure we are speaking to the correct boot load
er
    local major = execute(STK_GET_PARAMETER, 0x81, 1);
    local minor = execute(STK_GET_PARAMETER, 0x82, 1);
    local invalid = execute(STK_GET_PARAMETER, 0x83, 1);
    local signature = execute(STK_READ_SIGN);
    assert(major.len() == 1 && minor.len() == 1);
    assert((major[0] >= 5) || (major[0] == 4 && minor[0] >= 4));
    assert(invalid.len() == 1 && invalid[0] == 0x03);
    assert(signature.len() == 3 && signature[0] == 0x1E && signature[1] == 0x95 && signa
ature[2] == 0x0F);
}

//-----
function program_duino(address16, data) {

    local addr8_hi = (address16 >> 8) & 0xFF;
    local addr8_lo = address16 & 0xFF;
    local data_len = data.len();

    execute(STK_LOAD_ADDRESS, [addr8_lo, addr8_hi], 0);
    execute(STK_PROG_PAGE, [0x00, data_len, 0x46, data], 0)
    local data_check = execute(STK_READ_PAGE, [0x00, data_len, 0x46], data_len)

    assert(data_check.len() == data_len);
    for (local i = 0; i < data_len; i++) {
        assert(data_check[i] == data[i]);
    }
}

//-----
function bounce() {

    // Bounce the reset pin
    server.log("Bouncing the Arduino reset pin");
    RESET.configure(DIGITAL_OUT); //We need control of the reset pin

```

```
    imp.sleep(0.5);
    ACTIVITY.write(0); //Turn on LED
    RESET.write(0); //Reset Arduino
    imp.sleep(0.2);
    RESET.write(1); //Return reset to high, bootloader on Arduino now begins
    imp.sleep(0.3);
    check_duino();
    ACTIVITY.write(1); //Turn off LED

    RESET.configure(DIGITAL_IN); //Give up control of reset pin or else the board seems
to reset on each wakeup

}

//-----
function scan_serial() {
    local ch = null;
    local str = "";
    do {
        ch = SERIAL.read();
        if (ch != -1 && ch != 0) {
            str += format("%c", ch);
        }
    } while (ch != -1);

    if (str.len() > 0) {
        server.log("Serial: " + str);
    }
}

//-----
function burn(pline) {

    if ("first" in pline) {
        server.log("Starting to burn");
        SERIAL.configure(115200, 8, PARITY_NONE, 1, NO_CTSRTS);
        bounce();
    } else if ("last" in pline) {
        server.log("Done!")
        agent.send("done", true);
        SERIAL.configure(9600, 8, PARITY_NONE, 1, NO_CTSRTS, checkWeather);
    } else {
        program_duino(pline.addr, pline.data);
    }
}

//-----
```

```
//Everything above is used for bootloading

function toggleTxLED()
{
  txLEDToggle = 1 - txLEDToggle;    // toggle the txLEDToggle variable
  ACTIVITY.write(txLEDToggle);    // TX LED is on pin 8 (active-low)
}

function toggleRxLED()
{
  rxLEDToggle = 1 - rxLEDToggle;    // toggle the rxLEDToggle variable
  LINK.write(rxLEDToggle);    // RX LED is on pin 8 (active-low)
}

//When the agent detects a midnight cross over, send a reset to arduino
//This resets the cumulative rain and other daily variables
agent.on("sendMidnightReset", function(ignore) {
  server.log("Device midnight reset");
  SERIAL.write("@"); //Special midnight command
});

// Send a character to the Arduino to gather the latest data
// Pass that data onto the Agent for parsing and posting to Wunderground
function checkWeather() {

  //Get all the various bits from the Arduino over UART
  server.log("Gathering new weather data");

  //Clean out any previous characters in any buffers
  SERIAL.flush();

  //Ping the Arduino with the ! character to get the latest data
  SERIAL.write("!");

  //Wait for initial character to come in
  local counter = 0;
  local result = NOCHAR;
  while(result == NOCHAR)
  {
    result = SERIAL.read(); //Wait for a new character to arrive

    imp.sleep(0.01);
    if(counter++ > 200) //2 seconds
    {
      server.log("Serial timeout error initial");
      return(0); //Bail after 2000ms max wait
    }
  }
  //server.log("Counter: " + counter);
```

```
// Collect bytes
local incomingStream = "";
while (result != '\n') // Keep reading until we see a newline
{
    counter = 0;
    while(result == NOCHAR)
    {
        result = SERIAL.read();

        if(result == NOCHAR)
        {
            imp.sleep(0.01);
            if(counter++ > 20) //Wait no more than 20ms for another character
            {
                server.log("Serial timeout error");
                return(0); //Bail after 20ms max wait
            }
        }
    }

    //server.log("Test: " + format("%c", result)); // Display in log window

    incomingStream += format("%c", result);
    toggleTxLED(); // Toggle the TX LED

    result = SERIAL.read(); //Grab the next character in the que
}

server.log("We heard: " + format("%s", incomingStream)); // Display in log window
server.log("Arduino read complete");

ACTIVITY.write(1); //TX LED off

// Send info to agent, that will in turn push to internet
agent.send("postToInternet", incomingStream);

//imp.wakeup(10.0, checkWeather);
}

//These are needed for the wireless reprogramming
agent.on("burn", burn);
agent.send("ready", true);

SERIAL.configure(9600, 8, PARITY_NONE, 1, NO_CTSRTS); // 9600 baud worked well, no parity, 1 stop bit, 8 data bits

// Start this party going!
checkWeather();
```

```
//Power down the imp to low power mode, then wake up after 10 seconds
//Wunderground has a minimum of 2.5 seconds between Rapidfire reports
imp.onidle(function() {
  server.log("Nothing to do, going to sleep for 60 seconds");
  server.sleepfor(60);
});
```

