Universidad de La Laguna

ESCUELA POLITÉCNICA SUPERIOR DE INGENIERÍA
Sección Náutica, Máquinas y Radioelectrónica Naval

Trabajo presentado para la obtención del título de:

GRADUADO EN TECNOLOGÍAS MARINAS

Presentado por

Francisco Garoé Martín Ferreira

Prototipo orientado a su ejecución en un hardware libre para el monitoreo y control de parámetros de funcionamiento de un motor de combustión interna marino

Dirigido por

Carlos Efrén Mora Luis

Presentado en Septiembre 2018

Universidad de La Laguna

Resumen

El presente trabajo pretende crear una solución a los problemas detectados en el software que se utiliza para controlar y monitorizar los parámetros de un motor marino MAK 12VM 32C. Para ello, se recurre al uso de una plataforma de hardware y software libre denominada Arduino, que con una serie de periféricos conectados a sus entradas monitoriza y controla las condiciones de trabajo del motor. El sistema, a la vez que muestra los datos procesados de las condiciones de operación del motor, ejecuta las acciones necesarias para respetar las especificaciones establecidas por el fabricante. Este trabajo demuestra la viabilidad de usar Arduino para este tipo de aplicaciones y presenta posibles mejoras destinadas a mejorar el funcionamiento del motor.

Abstract

This project is aimed to provide an alternative solution to all the detected problems in the software used for controlling and monitoring the marine applicated engine's, MAK 12VM 32C, parameters. With that purpose, an open source hardware and software platform, called Arduino, is used. This device controls and watches the engine's working conditions due to using the peripherals connected to its inputs. The system shows the engine operational conditions processed data, at same time that carries out the necessary actions to respect the technical specifications stablished by the manufacturer as temperature of the cooling water. This project is intended to show the viability of using Arduino, besides of leaving an open door for future improvements, which optimise engine use.

Índice general

Índice de figuras

Índice de tablas

Simbología

Ω	Ohmios.
$^{\circ}C$	Grados centígrados.
mA	Miliamperios.
V	Voltios.

Acrónimos

NO Normalmente abierto

NC Normalmente cerrado

HT *High Temperature*

LT *Low Temperature*

NTC *Negative Temperature Coefficient*

FO Fuel Oil

SPI *Serial Peripheral Interface*

SPDT *Single Pole Double Throw*

LCD *Liquid Crystal Display*

PWM *Pulse-Width Modulation*

LED *Light Emitting Diode*

1 Introducción

El presente trabajo se plantea con el objetivo de desarrollar una alternativa eficaz, económica y sencilla que permita el control y la supervisión de un motor MAK 12VM32C prestando especial atención a los parámetros de funcionamiento que generan alarmas, paradas y reducciones de carga. Al mismo tiempo, esta propuesta trata de facilitar la localización de problemas de funcionamiento en el motor debido a parámetros inadecuados (p. ej. temperatura o presión).

La motivación de este trabajo surge a partir de la experiencia adquirida en las prácticas realizadas en el buque "OPDR CANARIAS" (noviembre 2017). Durante el periodo de embarque tuvieron lugar una serie de fallos relacionados con el software y las tarjetas de control que forman parte del sistema que monitoriza y controla el motor principal del buque.

Figura 1.1: Módulo de conexiones de las tarjetas de control y cuadro de gobierno de velocidad.

Fuente: Imagen de elaboración propia.

El problema con las tarjetas surgió después de una caída de la planta eléctrica del buque, la información del estado del motor se transmitía de manera errónea al monitor de diagnóstico.

El proyecto se centra en buscar una solución basada en una electrónica más actual como una alternativa al sistema de control existente. Para la realización de las tareas de monitorización y control del motor se ha propuesto un sistema electrónico de bajo coste y fácil manejo. Dicho sistema cuenta con un microcontrolador atmega2560 instalado en una placa Arduino Mega, que aprovecha los sensores preexistentes. La idea principal del trabajo es confirmar la viabilidad del proyecto mediante la realización de un montaje demostrativo, que ejecute un conjunto de simulaciones para verificar el correcto funcionamiento del sistema.

Se ha optado por un trabajo teórico-práctico en el se describen las herramientas utilizadas en el proyecto y se exponen los aspectos teóricos que facilitan la compresión del sistema, sin ahondar en la programación de Arduino. A continuación, se detallan las fases seguidas para el desarrollo del proyecto:

- **Investigación**. En esta etapa se ha hecho un estudio de los sensores que posee el motor y de los parámetros que son importantes para la seguridad del sistema. El estudio se ha hecho a partir de los manuales que proporciona el fabricante MAK con el fin de conseguir un funcionamiento óptimo de la máquina.
- **Elección y descripción de elementos:** Hardware utilizado, lenguaje de programación para el microcontrolador Arduino y esquemas de montaje del circuito.
- **Pruebas de funcionamiento y montaje final.** Evaluación del funcionamiento de los subsistemas mediante pruebas bien de manera física, o bien mediante la modificación del código fuente del sistema. En la demostración del sistema se pretende reproducir condiciones relevantes para el motor (p. ej. alta temperatura del agua de refrigeración o baja presión de aceite).

El trabajo se divide en 4 capítulos.

- Capítulo 1, Introducción.
- Capítulo 2, Fundamentación teórica, donde se describen los principios teóricos que sustenta el trabajo.
- Capítulo 3, Metodología, que describe el procedimiento seguido para desarrollar el trabajo.
- Capítulo 4, Software, relacionado con el software utilizado.
- Capítulo 5, Conclusiones.

2 Fundamentación teórica

Para facilitar las tareas de supervisión es necesario utilizar elementos que ayuden en la medición de las variables físicas más importantes como la presión o la temperatura. Con ese fin, el motor principal incorpora varios sensores y dispositivos distribuidos en los puntos críticos para su funcionamiento. Para garantizar un funcionamiento seguro del propulsor se han de mantener las variables físicas (presiones, temperaturas, etc.) dentro de en unos márgenes adecuados.

A continuación se expone una breve explicación del principio de funcionamiento de los sensores y dispositivos encargados de realizar la lectura de las variables físicas.

2.1 Sensores de presión

La monitorización de las presiones en el motor se hace a través de transductores y presostatos. Un transductor de presión absorbe la energía creada por la presión y la convierte en una señal eléctrica con salidas en voltaje o corriente. Los dispositivos de los que dispone el motor son de dos cables con una señal de corriente en miliamperios (4-20mA). Esta señal de salida varía linealmente con la presión [3].

Los transductores de presión son los encargados de activar las alarmas y las prealarmas. Existe una gran variedad de estos dispositivos, que en el caso del motor objeto de este trabajo, consisten en interruptores eléctricos movidos por un diafragma o un tubo de Bourdon. Los tubos de Bourdon consisten en un tubo hueco, y cilíndrico, enrollado en espiral y cerrado por uno de sus extremos. El funcionamiento de este dispositivo se basa en la deformación del tubo bajo la acción de una presión que deformará la espiral transmitiendo el movimiento [9] [24].

En el caso de los presostatos, se trata de un microinterruptor controlado por presión, que actúa a un determinado valor de dicha presión. Dentro de la extensa gama de presostatos se utilizan los de tipo diafragma (ver fig. 2.1). Este tipo de presostato funciona gracias a un elemento sensible y deformable (membrana) que se desplaza por una variación de presión, actuando sobre un microinterruptor a través de un vástago. Los contactos de los interruptores pueden ser de dos tipos:

- Normalmente abierto (NO) ó 1-2, siendo empleados para alarmas.

- Normalmente cerrado (NC) ó 3-4, siendo empleados para provocar paradas.

Con el fin de facilitar la interpretación de funcionamiento de un presostato de diafragma a continuación se muestra una imagen con su estructura.

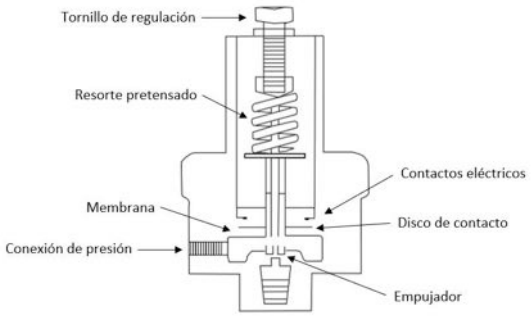

Figura 2.1: Presostato de diafragma.

Fuente: Imagen de elaboración propia.

Para las acciones de reducción de carga del motor, paradas y arranque de bombas en *standby* se opta por los presostatos. Para entender la importancia la variable de presión en el motor observemos en la siguiente lista los parámetros que controla:

- Aceite
- Aire de arranque y parada
- Combustible a la entrada del motor
- Presión diferencial en los filtros de combustible
- Presión del agua de refrigeración de alta temperatura, *High Temperature* (HT).
- Presión del agua de refrigeración de baja temperatura, *Low Temperature* (LT).

2.2 Sensores de temperatura

La medición de temperaturas en el motor se realiza por medio de termorresistencias PT100, sensores de temperatura NTC y termopares.

2.2.1 Termorresistencias PT100

Una termorresistencia PT100 es un sensor de temperatura basado en el aumento de la resistencia de un metal con el aumento de la temperatura. Sin embargo, el incremento de

la resistencia en la PT100 no es lineal, pero sí creciente aunque depende de los metales utilizados [6]. La característica más destacable de este tipo de termorresistencias es que están fabricadas de platino que tiene como característica que a 0(°C) tiene 100 ohms de resistencia.

El fabricante de motores MAK realiza el conexionado de la PT100 mediante tres hilos, cada uno de estos hilos tiene la misma resistencia eléctrica y están conectados a un puente de Wheatstone. Este conexionado elimina el problema de los errores generados por las grandes longitudes de cable que causan cambios de resistencia en la líneab[6] [4] [2]. Esto se consigue gracias a que el puente de Wheatstone originara una compensación y evitando así cambios en la señal eléctrica de la termorresistencia.

2.2.2 Sensores

Un sensor de temperatura *Negative Temperature Coefficient* (NTC) es un termistor con coeficiente de temperatura negativo respecto a la variación de su resistencia. Esto quiere decir que cuanto mayor sea la temperatura, menor será su resistencia. La variación de la resistencia con la temperatura en este tipo de sensores no es lineal, sino exponencial, y responde a la siguiente fórmula [23] [32] [30]:

$$R_T = R_o \epsilon^{B*(\frac{1}{T}-\frac{1}{T_o})} \tag{2.1}$$

Donde:

- $R_t \rightarrow$ Resistencia a temperatura T en grados Kelvin.
- $R_o \rightarrow$ Resistencia de referencia a una temperatura T_o en grados Kelvin.
- $B \rightarrow$ Valor B de la NTC.
- $T_o \rightarrow$ Temperatura de referencia en grados Kelvin.
- $T \rightarrow$ Temperatura a medir.

2.2.3 Termopares

Los termopares son sensores para la medición de temperatura basados en el principio de Seebeck. La unión de dos hilos de metales distintos para crear un circuito cerrado causa una corriente eléctrica cuando existe una diferencia de temperatura entre las uniones de ambos hilos. Existen varios tipos de termopares según la combinacion de metales que permiten la medida de un rango determinado de temperatura. El motor MAK objeto de este trabajo utiliza termopares tipo K cuyas características se exponen a continuación [21] [8] [27]:

- Rango de temperatura (-1200 a 1260°C)
- Rango de señal de -5.9mV a 51mV
- Límite de error ±2,2°C o ±0,75 %

Además la tensión de salida del termopar puede representarse como:

$$\epsilon = \alpha(T_2 - T_1) \tag{2.2}$$

Donde,

- $\alpha \rightarrow$ constante del termopar en mV/°C
- $\epsilon \rightarrow$ FEM (fuerza electro-motriz) por efecto Seebeck en mV
- $T_1, T_2 \rightarrow$ Temperaturas de cada unión en °C

T_1 se puede fijar como una unión de referencia y T_2 puede ser considerado como la unión con la temperatura medida. La siguiente gráfica muestra la respuesta del termopar.

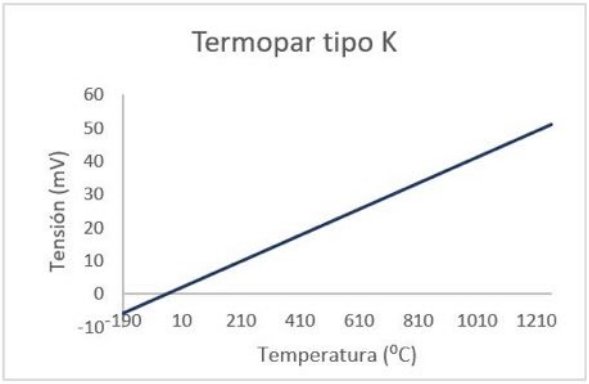

Figura 2.2: Temperatura frente a tensión termopar tipo K.

Fuente: Imagen de elaboración propia.

2.3 Sensor de velocidad

Para obtener la velocidad del motor MAK 12VM 32C se recurre a los sensores inductivos de tipo pick-up, ideales para detectar materiales ferromagnéticos. El sensor funciona gracias al principio inductivo-magnético, midiendo la frecuencia que genera la tensión que produce la bobina cuando se somete a un cambio de inductancia. Los cambios en las líneas de flujo ocasionan a la salida del sensor una señal de onda cuadrada [14] [18].

La bobina se encuentra arrollada en un imán que le proporciona un campo magnético fijo. Cuando un blanco de material ferromagnético se acerca al sensor provoca una disminución de la amplitud del campo electromagnético y la tensión disminuye, mientras que cuando

lo abandona la tensión aumenta. La tensión que produce la bobina varía en función de la velocidad con la que se introduce el blanco ferromagnético en el campo del imán [31] [20].

En la siguiente ilustración puede observar cómo el sensor detecta la rotación los dientes de un engranaje.

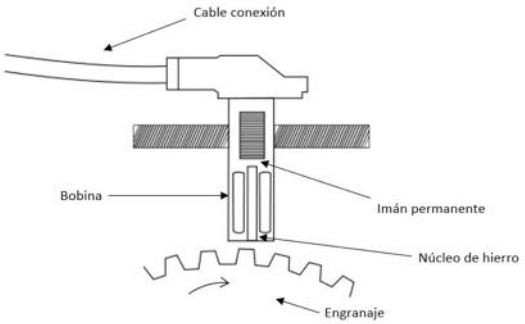

Figura 2.3: Sensor pick-up aplicado en un engranaje.

Fuente: Imagen de elaboración propia.

2.4 Sensor de detección de líquidos

La medición de los niveles de líquidos se realiza a través de sensores de nivel capacitivos, estos utilizan la diferencia de capacitancia entre un electrodo colocado en el mismo líquido y la superficie metálica del tanque o una sonda de referencia. Cualquier variación del nivel de líquido provoca una variación de la capacidad eléctrica que mediante el uso de un circuito de evaluación a la salida del sensor [7] [29] [13]. La siguiente imagen muestra un ejemplo del sensor aplicado en un tanque.

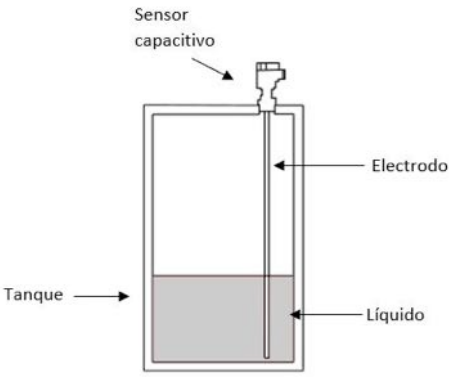

Figura 2.4: Sensor detector de líquidos.

Fuente: Imagen de elaboración propia.

2.5 Sensor de detección de posición

Para el ajuste de las bombas de combustible se utiliza un sensor de proximidad inductivo que detecta la posición de una leva acoplada al eje de la barra de inyección. Este tipo de sensores emiten un campo de detección electromagnético alterno, cuando un objeto metálico se aproxima se inducen corrientes de Foucault en el objetivo, en consecuencia, se reduce la amplitud de la señal que induce a un cambio de estado en la salida del sensor [3] [5] [22].

La señal de salida se acondiciona para lograr una salida de 4-20 mA. La siguiente imagen muestra como MAK ha implementado este sensor:

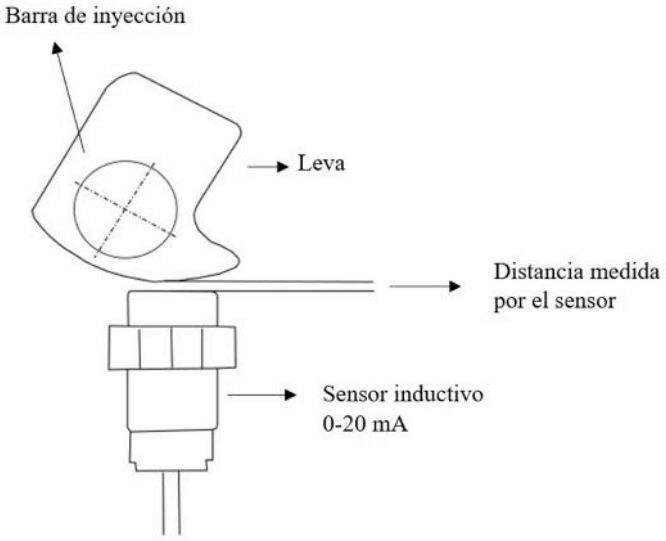

Figura 2.5: Aplicación de sensor inductivo en leva de posición.

Fuente: Imagen de elaboración propia.

2.6 Sensor detector de niebla del cárter

El sensor detector de niebla en el cárter, que MAK emplea, basa su funcionamiento en el método de extinción fotométrica (Ley Lambert-Beer). Para detectar la concentración de niebla, el sensor utiliza una muestra de la mezcla gaseosa proveniente del cárter. La muestra se succiona a través de un tubo conectado al cárter y, luego se conduce a través de un canal óptico. Si la absorción del rayo de luz aumenta quiere decir que existe una mayor turbidez y una mayor opacidad, si el porcentaje de opacidad de la muestra gaseosa supera el valor establecido se activará la alarma [28] [1] [12].

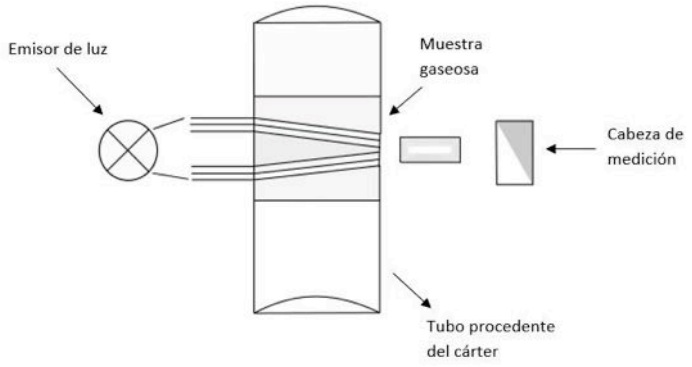

Figura 2.6: Sensor detector de niebla en el cárter.

Fuente: Imagen de elaboración propia.

3 Metodología

3.1 Determinación de las variables de supervisión del motor de combustión interna

Las variables de supervisión son aquellas magnitudes físicas correspondientes a un sistema controlado, de forma que su valor no exceda ciertos límites dentro de su operación normal. Cuando alguna de estas variables supera su límite preestablecido, se deben tomar medidas que salvaguarden la integridad y la seguridad de la planta. En el caso del motor que se ha decidido supervisar, objeto de este proyecto, se han tomado los mismos parámetros monitorizados por el sistema de supervisión preexistente, a partir de las especificaciones técnicas del fabricante del motor especificadas en la siguiente tabla [3]:

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
1	Transmisor de presión Baja presión de aceite Prealarma reducción de carga	0-600 kPa 4-20 mA	500 kPa	305 kPa
2	Interruptor de presión Baja presión de aceite Reducción de carga	0-600 kPa Todo/nada	500 kPa	300 kPa
3	Interruptor de presión Baja presión de aceite Arranca bomba en stdby	0-600 kPa Todo/nada	500 kPa	330 kPa
4	Interruptor de presión Baja presión de aceite Parada	0-600 kPa Todo/nada	500 kPa	270 kPa

continúa en la página siguiente

Continúa de la página anterior

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
5	Interruptor de presión Baja presión de agua HT a la entrada del motor Arranca bomba en stdby	0-600 kPa Todo/nada	> 250 kPa	20 kPa por debajo del valor de funcionamiento
6	Interruptor de presión Baja presión de agua HT a la entrada del motor Alarma	0-600 kPa Todo/nada	> 250 kPa	40 kPa por debajo del valor de funcionamiento
7	Interruptor de presión Baja presión de agua HT a la entrada del motor Parada	0-600 kPa Todo/nada	> 250 kPa	60 kPa por debajo del valor de funcionamiento
8	Interruptor de presión Baja presión de agua LT a la salida del motor Arranca bomba en stdby	0-600 kPa Todo/nada	> 250 kPa	20 kPa por debajo del valor de funcionamiento
9	Interruptor de presión Baja presión de agua LT a la salida del motor Alarma	0-600 kPa 4-20 mA	> 200 kPa	40 kPa por debajo del valor de funcionamiento
10	Interruptor de presión Baja presión de Fuel Oil (FO) a la entrada del motor Arranca bomba en stdby	0-600 kPa Todo/nada	350 kPa	280 kPa
11	Interruptor de presión Baja presión de FO a la entrada del motor Alarma	0-1000 kPa 4-20 mA	400 kPa	100 kPa
12	Interruptor presión diferencial Alta presión diferencial filtros FO Alarma	0-80 kPa Todo/nada	< 60 kPa	80 kPa
13	Transmisor de presión Baja presión de aire de arranque Alarma	0-6000 kPa 4-20 mA	< 3000 kPa	1250 kPa
14	Interruptor de presión Baja presión de aire de parada Alarma	100-1200 kPa	2s 750 kPa	600 kPa
15	Interruptor de presión Presión de aire de arranque en la válvula de arranque principal Alarma	1-1600 kPa Todo/nada	> 1200 kPa	1000 kPa
16	Alta temperatura de aceite a la entrada del motor Alarma	-40 - 160 °C PT 100	60-65 °C	70 °C
17	Alta temperatura de aceite a la entrada del motor Parada	-40 - 120 °C NTC	60-65 °C	75 °C

continúa en la página siguiente

Continúa de la página anterior

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
18	Termorresistencia Temperatura de agua de refrigeración HT a la entrada del motor DICARE	-40 - 160 °C PT 100	80 - 90 °C	60 °C
19	Termorresistencia Alta temperatura de agua de refrigeración HT a la salida del motor Alarma	-40 - 160 °C PT 100	80 - 90 °C	93 °C
20	Alta temperatura agua de refrigeración HT a la salida del motor Reducción	-40 - 120 °C NTC	80 - 90 °C	98 °C
21	Termorresistencia Temperatura de agua de refrigeración LT a la entrada del motor Alarma	-40 - 160 °C PT 100	38 °C	43 °C
22	Termorresistencia Baja o alta temperatura FO a la entrada del motor Alarma	-40 - 160 °C PT 100	40 - 50 °C	35 - 55 °C
23	Alta temperatura del aire de carga a la entrada del motor Alarma	-40 - 160 °C PT 100	40 - 50 °C	65 °C
24	Termopar tipo K Temperatura de salida de gases de escape cilindros 1A...6A Reducción de carga	- 200 - 1300 °C NiCr-Ni (mV)	350 / 450 °C	490 °C
25	Termopar tipo K Temperatura de salida de gases de escape cilindros 1B...6B Reducción de carga	- 200 - 1300 °C NiCr-Ni (mV)	350 / 450 °C	490 °C
26	Termopar tipo K Temperatura de salida de gases de escape salida tubo A	- 200 - 1300 °C NiCr-Ni (mV)	480 °C - 25 % 380 °C - 100 %	>400 °C >500 °C
27	Termopar tipo K Temperatura de salida de gases de escape salida tubo B	- 200 - 1300 °C NiCr-Ni (mV)	480 °C - 25 % 380 °C - 100 %	>400 °C >500 °C
28	Termorresistencia Control eléctrico de temperatura de aire de carga		-40 - 160 °C PT 100	
29	Pick-up Velocidad del motor	0-1200 upm 0-15 kHz	720 - 750 rpm	<0.7 N nominal
30	Pick-up Velocidad regulador electrónico			

continúa en la página siguiente

Continúa de la página anterior

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
31	Pick-up Medidor de revoluciones turbo A	0 - 40000 upm 0 - 1 kHz 0 - 10 V	<30000 rpm	30000 rpm
32	Pick-up Medidor de revoluciones turbo B	0 - 40000 upm 0 - 1 kHz 0 - 10 V	<30000 rpm	30000 rpm
33	Sonda de nivel Alto nivel de aceite en la bandeja del motor Alarma	Todo/nada	Contactor - NC	
34	Sonda de nivel Condensado de agua en el canal de aire Alarma	Todo/nada	Contactor - NC	
35	Sensor de posición Ajuste de la bomba de combustible	0-125 % de Carga 0-70 mm	Salida 4-20 mA	
36	Detector de niebla del cárter Alta concentración de neblina de aceite en la bandeja del cárter	0,5...25 % Todo/nada	<2 %	>2 %

Tabla 3.1: Lista de señales de control del motor MAK 12VM32C.

3.2 Análisis y adaptación de los protocolos de medida y comunicación

3.2.1 Medida de temperaturas bajas

Para obtener los resultados de temperatura a través de las termorresistencias PT-100 (correspondientes a los sensores: 16, 18, 19, 21, 22, 23 de la tabla 3.2) y termistores NTC (sensores 17 y 20 de la tabla 3.2), resulta necesario convertir la señal que proporcionan de origen a una con salida 4-20 mA.

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
16	Termorresistencia Alta temperatura de aceite a la entrada del motor Alarma	-40 - 160 °C PT 100	60 - 65 °C	70 °C

continúa en la página siguiente

Continúa de la página anterior

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
17	NTC Alta temperatura de aceite a la entrada del motor Parada	-40 - 120 °C NTC	60 - 65 °C	75 °C
18	Termorresistencia Temperatura de agua de refrigeración HT a la entrada del motor Dicare	-40 - 160 °C PT 100	80 - 90 °C	60 °C
19	Termorresistencia Alta temperatura de agua de refrigeración HT a la salida del motor Alarma	-40 - 160 °C PT 100	80 - 90 °C	93 °C
20	NTC Alta temperatura de agua de refrigeración HT a la salida del motor Reducción	-40 - 120 °C NTC	80 - 90 °C	98 °C
21	Termorresistencia Temperatura de agua de refrigeración LT a la entrada del motor Alarma	-40 - 160 °C PT 100	38 °C	43 °C
22	Termorresistencia Baja o alta temperatura de Fuel Oil a la entrada del motor	-40 - 160 °C PT 100	40 - 50 °C	35 - 55 °C
23	Termorresistencia Alta temperatura del aire de carga a la entrada del motor Alarma	-40 - 160 °C PT 100	40 - 50 °C	65 °C

Tabla 3.2: Termorresistencias PT-100 y termistores NTC para medición de bajas temperaturas.

Tanto en las termorresistencias PT-100, como en los termistores NTC, se utiliza un convertidor linealizado basado en microprocesador que proporciona una señal de salida de 4-20 mA (señal estándar en instrumentación industrial). La corriente de 4-20 mA es posteriormente convertida en una señal de voltaje (0-5V) con el fin de poder utilizar una entrada analógica (A0) de la placa Arduino.

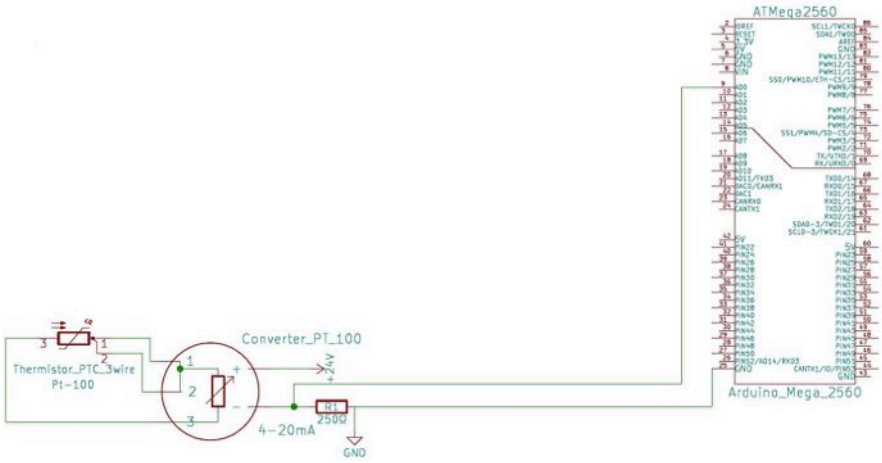

Figura 3.1: Esquema de conexionado del sensor PT 100.

Fuente: Imagen de elaboración propia.

Esta segunda transformación se fundamenta en el uso de la ley de Ohm ($V = R \cdot I$) se utiliza una resistencia de 250 Ohmios de forma tal que si la señal primaria es de en 4 mA le corresponde 1V, y cuando aumente a 20mA le corresponderán 5V. Para mantener las condiciones del termistor original, la alimentación del lazo se realizó con una fuente de 24V-DC.

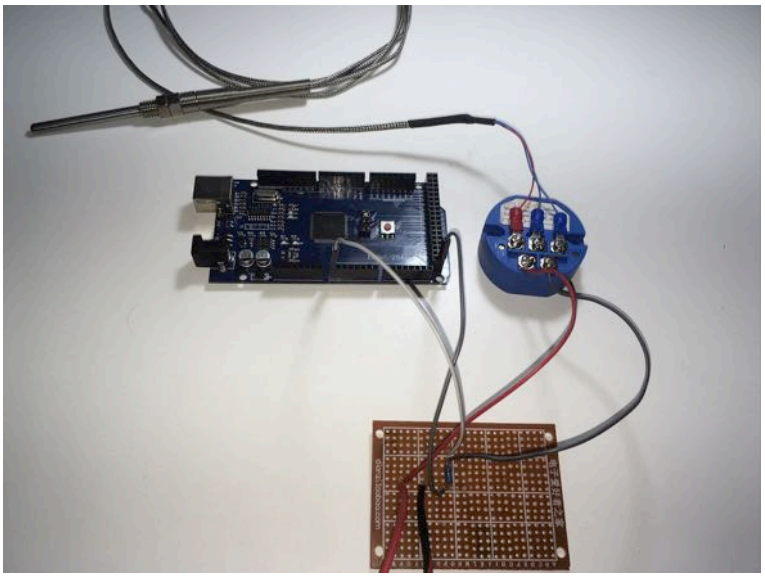

Figura 3.2: Montaje de sensor de temperatura PT-100.

Fuente: Imagen de elaboración propia.

Aunque ya es posible la lectura del sensor mediante el sistema, resulta necesario realizar un mapeo de los valores mínimos y máximos de la señal: el valor de 0°C corresponde a 1V para la entrada analógica de Arduino que lo identifica con un valor numérico de 205; el valor de 200°C, corresponde a 5V y un valor numérico de 1023.

3.2.2 Medida de temperaturas altas

Para la medida de temperaturas no se realiza solo por termistores, también se recurre a termopares tipo K (sensores 24, 25 ,26 y 27 de la tabla 3.3) que producen una pequeña tensión proporcional a la temperatura.

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
24	Termopar tipo K Temperatura de salida de gases de escape cilindros 1A...6A	-200 a 1300 °C NiCr-Ni (mV)	350 / 450 °C	490 °C
25	Termopar tipo K Temperatura de salida de gases de escape cilindros 1B...6B	-200 a 1300 °C NiCr-Ni (mV)	350 / 450 °C	490 °C
26	Termopar tipo K Temperatura de salida de gases de escape a la salida de turbo A	-200 a 1300 °C NiCr-Ni (mV)	480 °C - 25 % 380 °C - 100 %	>400 °C >500 °C
27	Termopar tipo K Temperatura de salida de gases de escape a la salida de turbo B	-200 a 1300 °C NiCr-Ni (mV)	480 °C - 25 % 380 °C - 100 %	>400 °C >500 °C

Tabla 3.3: Termopares tipo K medición de temperaturas altas.

Para una aplicación basada en microcontrolador, las señales de los termopares necesitan ser acondicionadas:

1. Resulta necesaria una amplificación de la señal de salida del termopar.

2. Linealizar la respuesta del termopar.

3. realizar una compensación de la unión fría.

Con el fin de acondicionar la señal para poder ser procesada por Arduino, se recurre a un módulo conversor analógico-digitalque permite compensar y convertir el voltaje creado por el termopar. El conversor utilizado dispone de un circuito integrado MAX 6675 fabricado por Maxim/Dallas.

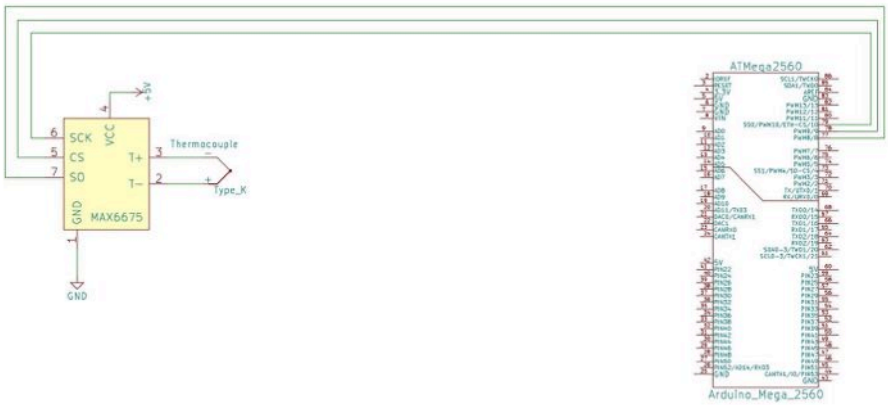

Figura 3.3: Comunicación módulo MAX 6675.

Fuente: Imagen de elaboración propia.

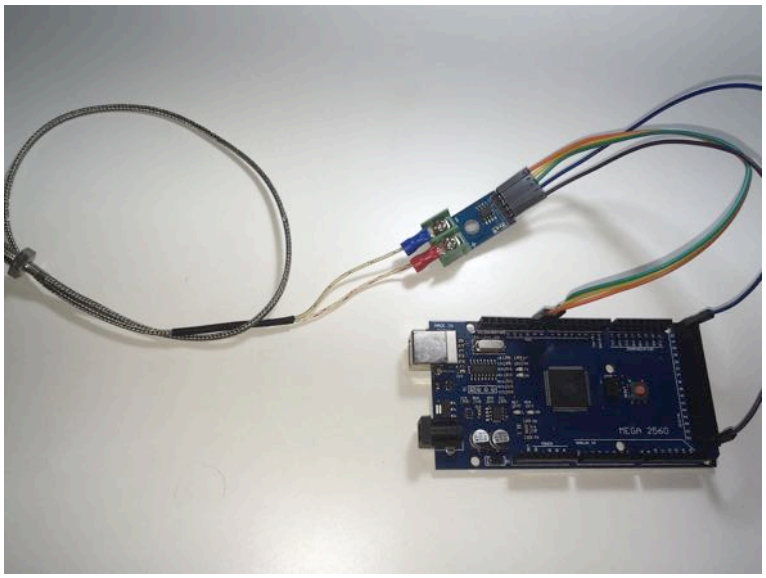

Figura 3.4: Pulsadores para simular señal presostatos.

Fuente: Imagen de elaboración propia.

El MAX6675 se conecta al Arduino mediante una interfaz de 3 líneas compatible con el estándar de comunicaciones *Serial Peripheral Interface* (SPI) empleado para la comunicación entre circuitos integrados y equipos electrónicos. El formato en el que el MAX6675 envía datos al microcontrolador es el siguiente: [11] [34] [15].

Bit	Dummy Sing Bit	12-Bit Temperature reading												Thermocouple input	Device ID	State
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		MSB											LSB		0	Three satate

Tabla 3.4: Formato que usa MAX6675 para el envío de datos al microcontrolador.

3.2.3 Simulación de la señal de los sensores

3.2.3.1 Simulación de las señales de presión

La presión es monitorizada a través de transductores (1, 9, 11, 13 de la tabla 3.5) y presostatos (2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 15 de la tabla 3.5). En cuanto a los transductores, estos ofrecen una salida analógica de 4-20 mA. En el caso de los presostatos, la señal de salida se realiza mediante un contacto doble unipolar *Single Pole Double Throw* (SPDT), que conmuta un polo común a otros dos polos [16].

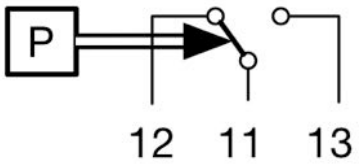

Figura 3.5: Salida de presostato de contacto doble unipolar.

Fuente: Imagen de elaboración propia.

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
1	Transmisor de presión Baja presión de aceite Prealarma reducción de carga	0-600 kPa 4-20 mA	500 kPa	305 kPa
2	Interruptor de presión Baja presión de aceite Reducción de carga	0-600 kPa Todo/nada	500 kPa	300 kPa

continúa en la página siguiente

Continúa de la página anterior

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
3	Interruptor de presión Baja presión de aceite Arranca bomba en stdby	0-600 kPa Todo/nada	500 kPa	330 kPa
4	Interruptor de presión Baja presión de aceite Parada	0-600 kPa Todo/nada	500 kPa	270 kPa
5	Interruptor de presión Baja presión de agua HT a la entrada del motor Arranca bomba en stdby	0-600 kPa Todo/nada	$>$ 250 kPa	20 kPa por debajo del valor de funcionamiento
6	Interruptor de presión Baja presión de agua HT a la entrada del motor Alarma	0-600 kPa Todo/nada	$>$ 250 kPa	40 kPa por debajo del valor de funcionamiento
7	Interruptor de presión Baja presión de agua HT a la entrada del motor Parada	0-600 kPa Todo/nada	$>$ 250 kPa	60 kPa por debajo del valor de funcionamiento
8	Interruptor de presión Baja presión de agua LT a la salida del motor Arranca bomba en stdby	0-600 kPa Todo/nada	$>$ 250 kPa	20 kPa por debajo del valor de funcionamiento
9	Interruptor de presión Baja presión de agua LT a la salida del motor Alarma	0-600 kPa 4-20 mA	$>$ 200 kPa	40 kPa por debajo del valor de funcionamiento
10	Interruptor de presión Baja presión de FO a la entrada del motor Arranca bomba en stdby	0-600 kPa Todo/nada	350 kPa	280 kPa
11	Interruptor de presión Baja presión de FO a la entrada del motor Alarma	0-1000 kPa 4-20 mA	400 kPa	100 kPa
12	Interruptor presión diferencial Alta presión diferencial filtros FO Alarma	0-80 kPa Todo/nada	$<$ 60 kPa	80 kPa
13	Transmisor de presión Baja presión de aire de arranque Alarma	0-6000 kPa 4-20 mA	$<$ 3000 kPa	1250 kPa
14	Interruptor de presión Baja presión de aire de parada Alarma	100-1200 kPa	2s 750 kPa	600 kPa

continúa en la página siguiente

Continúa de la página anterior

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
15	Interruptor de presión Presión de aire de arranque en la válvula de arranque principal Alarma	1-1600 kPa Todo/nada	> 1200 kPa	1000 kPa

Tabla 3.5: Presostatos y transductores utilizados para obtener los valores de presión.

En las pruebas de funcionamiento de este proyecto, para simular la señal de salida de los presostatos, se ha optado por el uso de pulsadores de contacto NO(ver figura 3.4). Para la simulación de las señales de 4-20 mA que emulan la señal de los transductores de presión, se ha construido un circuito, el cual ofrece la posibilidad de modificar la señal de salida y así reproducir diferentes escenarios de funcionamiento.

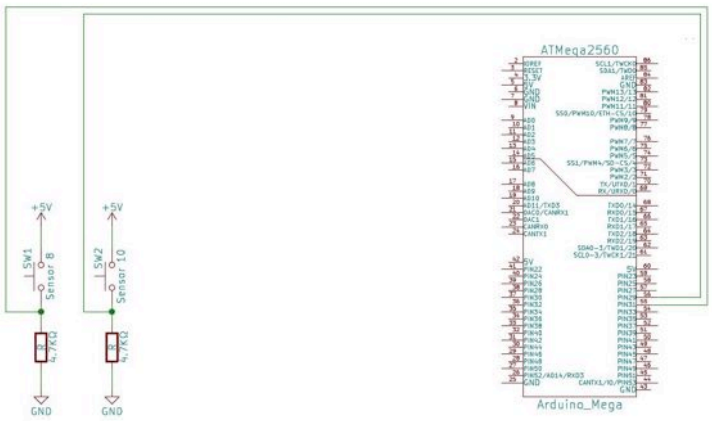

Figura 3.6: Pulsadores para simular señal presostatos.

Fuente: Imagen de elaboración propia.

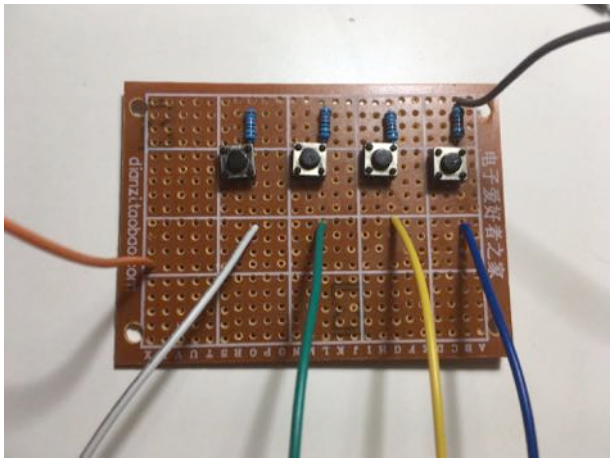

Figura 3.7: Montaje de pulsadores para simular la señal de los presostatos (imagen de elaboración propia).

El circuito para emular los transductores de presión usa un amplificador diseñado para mantener la corriente a un valor establecido (4 – 20 mA). Para ello se le aplica una señal de control de bajo voltaje (1 – 5V) al circuito de carga para variar la señal de salida. Un amplificador operacional con retroalimentación negativa recibe la señal de control y mantiene la intensidad de corriente que corresponda a cada caso.

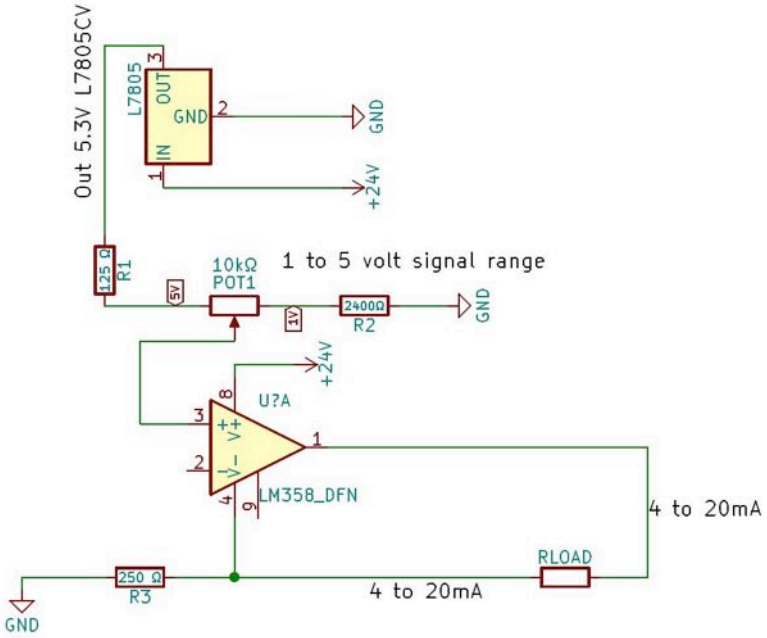

Figura 3.8: Conversión de señal de voltaje a corriente.

Fuente: Imagen de elaboración propia.

La tensión de entrada a este circuito se ha acondicionado mediante un regulador de voltaje lineal LM7805 que entrega una tensión de 5 voltios, por medio del uso de un potenciómetro calibrado mediante resistencias se produce una tensión al circuito de 1 voltio al 0 por ciento

de la medición física y 5 voltios al 100 por ciento. A una entrada de 5 voltios, la resistencia de 250Ω tendrá 5 voltios aplicados a través de ella, lo que resulta en 20 mA la relación entre el voltaje de entrada y la corriente de salida, en este caso es de 1-5 V en / 4-20 mA de salida [10].

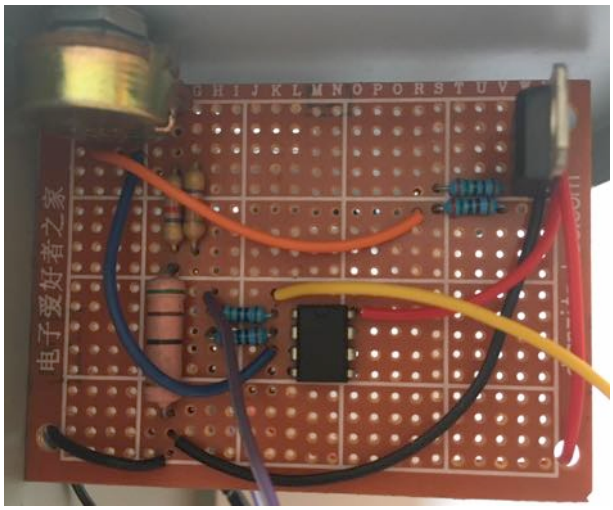

Figura 3.9: Montaje simulador 4-20 mA.

Fuente: Imagen de elaboración propia.

Resulta necesario realizar un mapeo de los valores mínimos y máximos de la señal: el valor de 0 kPa corresponde a 1V para la entrada analógica de Arduino que lo identifica con un valor numérico de 0; el valor de 600kPa, corresponde a 5V y un valor numérico de 1023.

3.2.3.2 Simulación de las señales de velocidad

La velocidad del motor y turbos se vigila mediante sensores inductivos de tipo pick-up (29, 30, 31, 32 de la tabla 3.6).

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
29	Pick-up Velocidad del motor	0-1200 upm 0-15 kHz	720- 750 rpm	<0.7 N nominal
30	Pick-up Velocidad regulador electrónico	—-	—-	—-
31	Pick-up Medidor de revoluciones turbo A	0-40000 upm 0-1 kHz 0-10V	<30000 rpm	30000 rpm
32	Pick-up Medidor de revoluciones turbo B	0-40000 upm 0-1 kHz 0-10V	<30000 rpm	30000 rpm

Tabla 3.6: Sensores de velocidad tipo pick-up.

Para la simulación de la señal del sensor, se ha optado por utilizar un sensor magnético efecto Hall A3144 con salida digital y retención de estado, diseñado para su uso en microcontroladores. El sensor Hall monitorea la velocidad de un motor eléctrico controlado mediante una salida PWM de la placa Arduino. Para producir el campo electromagnético necesario para que el sensor cambie de estado se instala en el eje del motor una polea con un imán en un extremo de su superficie.

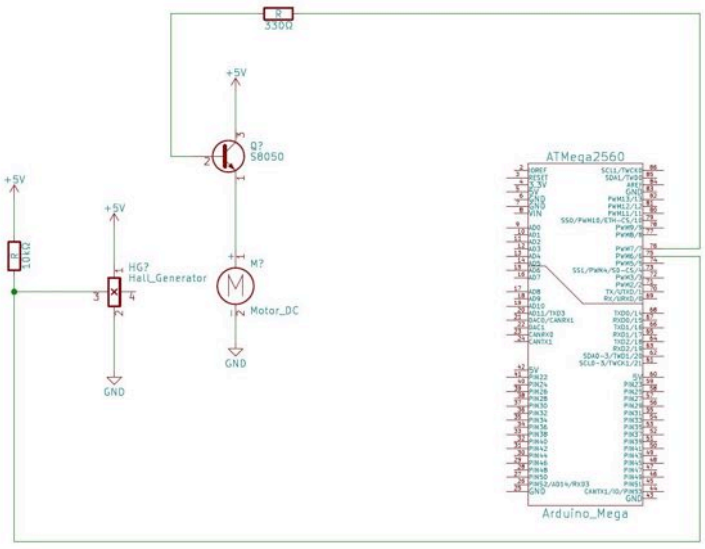

Figura 3.10: Esquema simulador sensores tipo pick-up.

Fuente: Imagen de elaboración propia.

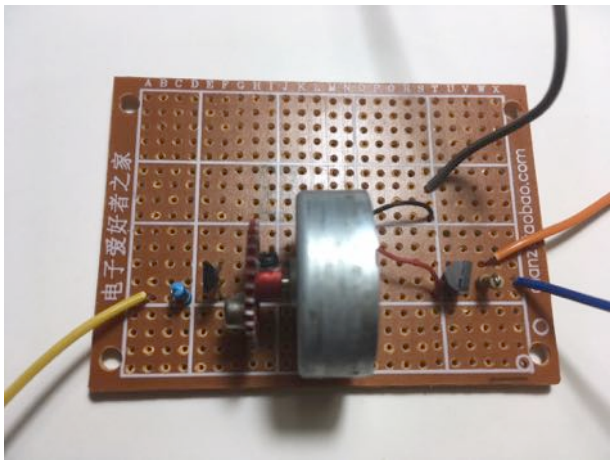

Figura 3.11: Montaje para simular la señal de velocidad de los sensores tipo pick-up.

Fuente: Imagen de elaboración propia.

3.2.3.3 Simulación de las señales de nivel

Para el control de los niveles de líquidos, el motor MAK objeto de este proyecto usa sensores capacitivos todo/nada (números 33 y 34 de la tabla 3.7).

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
33	Sonda de nivel Alto nivel de aceite bandeja motor Alarma	Todo/nada	Contactor -NC	
34	Sonda de nivel Condensado de agua en el canal de aire Alarma	Todo/nada	Contactor -NC	

Tabla 3.7: Sensores de nivel de tipo capacitivo.

La simulación de las señales de nivel se hace mediante pulsadores NO. Dichos pulsadores se conectan entradas digitales de la placa Arduino.

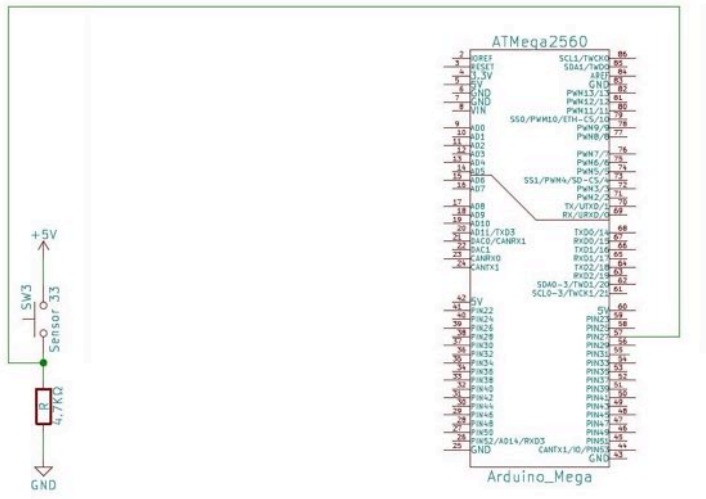

Figura 3.12: Pulsador para simular la señal del sensor de nivel.

Fuente: Imagen de elaboración propia.

3.2.3.4 Simulación de la señal del sensor de proximidad

Para detectar el ajuste de la bomba del combustible se emplea un de proximidad inductivo (35 de la tabla 3.8).

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
35	Ajuste de la bomba de combustible	0-125 % Carga 0-70mm	Salida 4-20mA	

Tabla 3.8: Sensor de distancia de tipo inductivo.

Los sensores de proximidad ofrecen una salida analógica de 4-20 mA. Con el fin de simular su funcionamiento se recurre al mismo montaje que en el caso de la simulación de la señal de salida de los transductores (ver figura 3.5).

3.2.3.5 Simulación de la señal del sensor detector de niebla en el cárter

Para detectar la presencia de niebla en el cárter, el motor objeto de este trabajo emplea un sensor (36 de la tabla 3.8) que dispone de una salida todo/nada.

Número del sensor	Descripción	Rango del sensor Rango de la señal	Valor de operación	Disparo
36	Detector de niebla del cárter Alta concentración de neblina de aceite bandeja del cárter	0,5..25 % Todo/nada	<2 %	>2 %

Tabla 3.9: Sensor detector de niebla en el cárter.

La simulación del sensor detector de niebla en el cárter se hace mediante un pulsador NO conectado a la entrada de la placa Arduino.

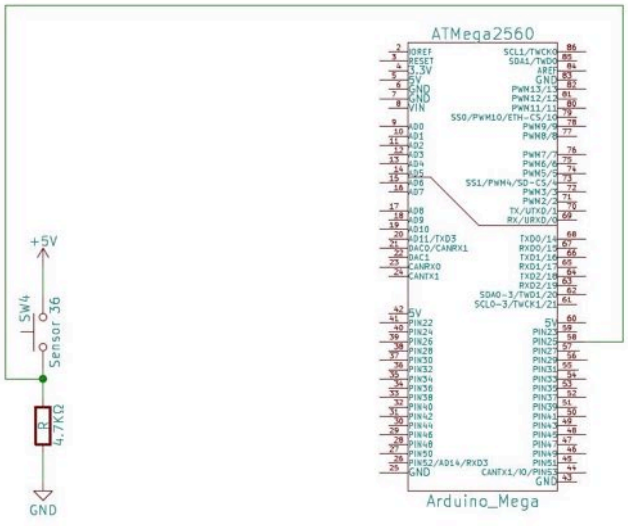

Figura 3.13: Pulsador para simular la señal del sensor de niebla.

Fuente: Imagen de elaboración propia.

3.3 Información al usuario del estado del motor

Atendiendo al listado de alarmas, paradas y reducciones de potencia proporcionado por el fabricante, se define el programa que permite el control del motor para mantener los rangos normales de operación.

Para proporcionar información al usuario sobre los parámetros (p. ej. temperatura o presión) actuales en el motor, se crea un programa que el operario puede navegar entre sus diferentes menús mediante el movimiento mecánico rotatorio de un encoder provisto de pulsador que realiza la función de "ok" o de "entrar en el menú".

A través de un panel LCD provisto de 16 caracteres y 2 líneas se muestran los parámetros en tiempo real, lo que permite conocer con exactitud el estado actual del motor. Cuando un parámetro se encuentre fuera de su valor normal de operacion se activarán diferentes señales luminosas y una señal sonora. La siguiente lista describe las acciones del programa:

- Parámetro por debajo del valor normal de operación → produce una alarma con señal acústica y luminosa mediante un led de color ámbar.
- Parámetro por dentro del valor normal de operación → origina una señal luminosa mediante un led verde.
- Parámetro por encima del valor normal de operación → Produce una alarma que puede conllevar una reducción de carga o parada del motor mediante una señal acústica y luminosa por medio de un led de color rojo.

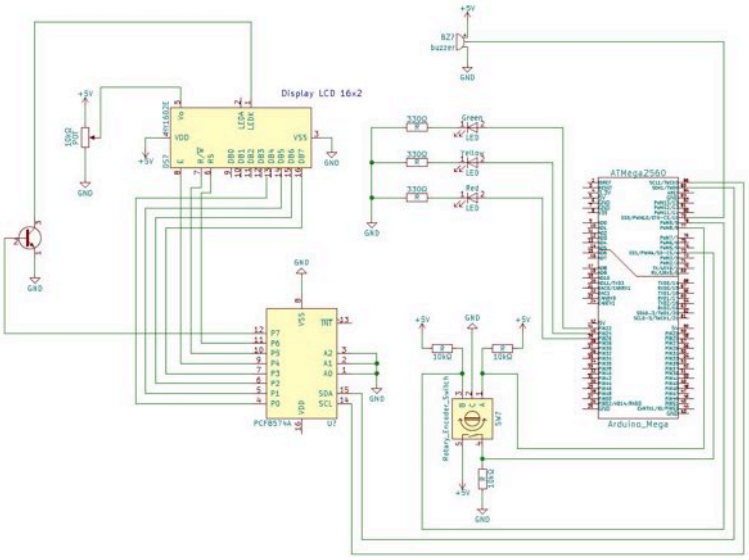

Figura 3.14: Módulo de interface para el usuario.

Fuente: Imagen de elaboración propia.

3.4 Desarrollo del software

Este proyecto se ha desarrollado en Arduino, se trata de una placa electrónica que tiene un microcontrolador programable y dispone de varias entradas y salidas que pueden ser analógicas ó digitales [17].

La gran ventaja de esta placa es que es de código abierto, tanto de hardware como de software. Las instrucciones se realizan a través de un IDE propio que utiliza un lenguaje C con algunas funciones de C++. La estructura básica del lenguaje consta de dos funciones principales: void setup(), encargada de recoger la configuración; y la segunda, void loop(), es la parte que contiene el programa, la cual se ejecuta de forma cíclica [25] [33].

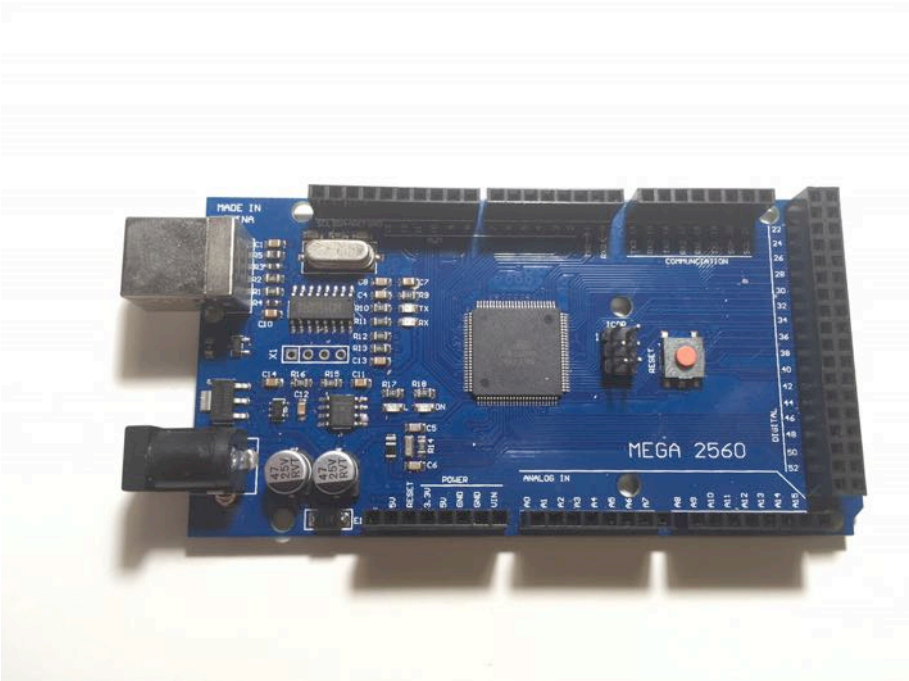

Figura 3.15: Vista de la placa Arduino MEGA 2560.

Fuente: Imagen de elaboración propia.

Las características de la placa utilizada son las siguientes:

- Microcontrolador ATmega2560.
- 54 pines digitales de entrada/salida, 14 de ellos son salidas *Pulse-Width Modulation* (PWM)
- 16 entradas análogas

3.4.1 Algoritmo de control

La figura 3.16 muestra la secuencia de deciciones que toma el programa para realizar las acciones necesarias y monitorear el estado del motor.

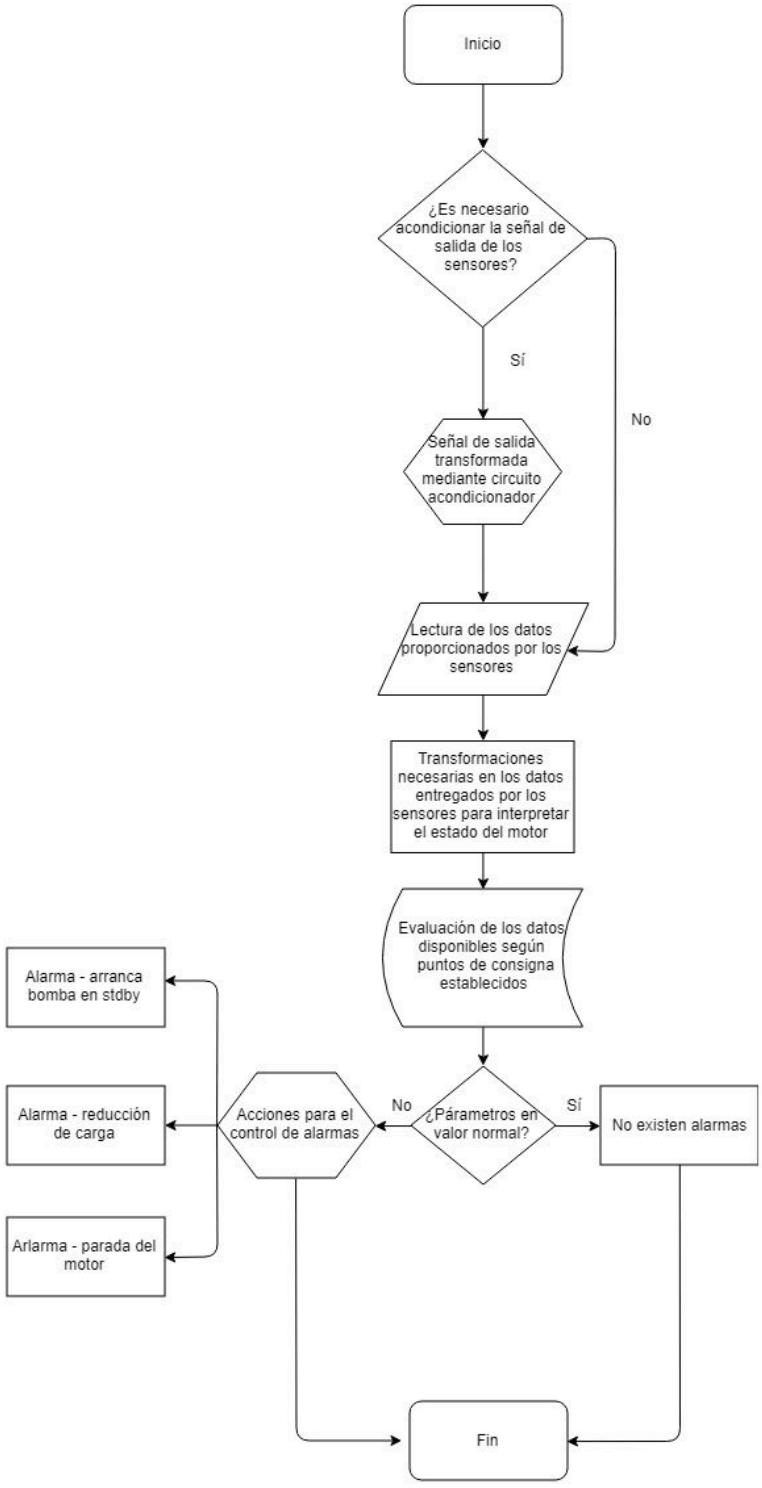

Figura 3.16: Diagrama de flujo del algoritmo del programa.

Fuente: Imagen de elaboración propia.

El funcionamiento de cada uno de los bloques del algoritmo de control es el siguiente:

- **¿Es necesario acondicionar la señal de salida de los sensores?** Aquellos sensores en los que su lectura mediante Arduino no se pueda realizar directamente es necesario utilizar un circuito acondicionador para trasnformar su señal de salida

como podemos observar en los apartados anteriores 3.2.1 y 3.2.2. (Ver líneas de código de 706 a 743).

- **Señal de salida transformada mediante el uso de circuito acondicionador.** Se encuentra disponible para su lectura mediante las entradas analógicas o digitales la señal de salida de los sensores que requieren de una adecuación. (Se lee la señal ya acondicionada en la líneas de código 727 a 736).

- **Lectura de los datos proporcionados por los sensores.** En este punto del bucle se lee el estado del motor leyendo los valores de las entradas analógicas y digitales asociadas a los sensores descritos en la sección 3.1. Dichos valores son almacenados en un juego de variables globales de forma que pueden ser accedidas por otras funciones. (Ver líneas de código 706 a 743).

- **Transformaciones necesarias en los datos entregados por los sensores para interpretar el estado del motor.** En el caso que fuera necesario transformar los datos aportados por los sensores (p. ej. apartado 3.2.1) se realiza mediante el código del programa una serie de operaciones matemáticas. (Ver líneas de código de 744 a 770).

- **Evaluación de los datos disponibles según puntos de consigna establecidos.** Teniendo en cuenta la lista de puntos de consigna suministrados por el fabricante y los datos entregados por los sensores se evalua el estado actual del motor. (Ver líneas de código de 744 a 770)

- **¿Parámetros en valor normal?** Dependiendo de los datos entregados por los sensores se emprenden las acciones para el control de alarmas. (Ver líneas de código de 326 a 419 y de 744 a 770).

- **Acciones para el control de alarmas.** Según el contraste que surge entre el estado actual del motor y los puntos de consigna establecidos se inician tres posibles estados de alarma. (Ver línea de código 749 y 767).

- **Alarma arranca bomba en stdby.** Comprende un estado de alarma menos grave, arranca una bomba en stdby como acción para compensar la desigualdad entre el estado de los parámetros motor y los valores normales de operación.

- **Alarma reducción de carga.** Comprende un estado de alarma grave, existe una desigualdad considerable entre los parámtros actuales del motor y los valores normales de operación, produce una reducción de carga del motor para garantizar su integridad. (Ver líneas de código de 757 a 763)

- **Alarma parada.** Comprende un estado de alarma muy grave, existe una gran desigualdad entre los parámtros actuales del motor y los valores normales de operación, produce la parada inmediata del motor para garantizar su integridad. (Ver línea de código 749).

- **No existen alarmas.** Los parámetros del motor operan dentro del rango normal de operación. (Ver líneas de código de 764 a 770).

3.4.2 Desarrollo de la interfaz de usuario

A continuación se explica la configuracion de los menús que se muestran a través de la pantalla *Liquid Crystal Display* (LCD) para que el usuario pueda acceder a la información e interactuar con el software de control (ver figura 3.17).

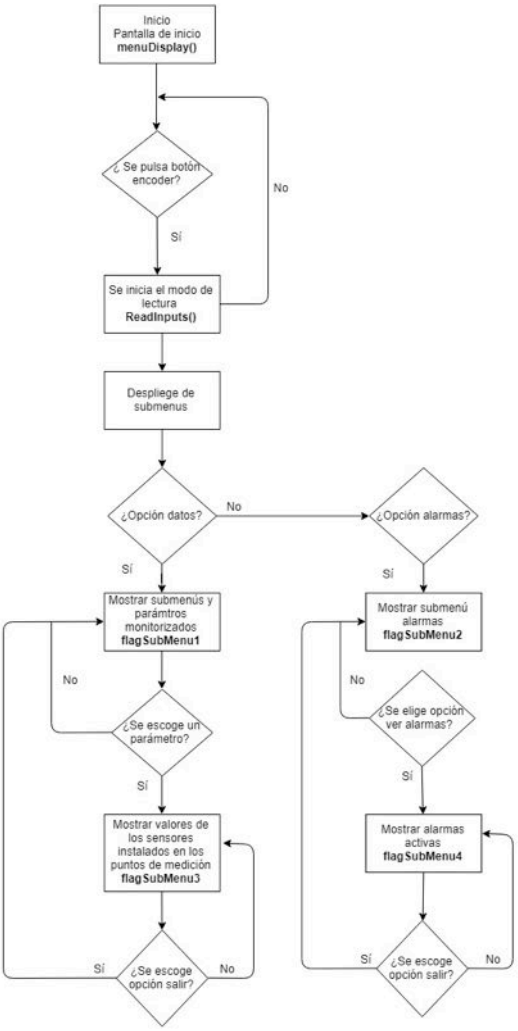

Figura 3.17: Diagrama de flujo iterface y menus.

Fuente: Imagen de elaboración propia.

3.4.2.1 Inicialización y menus

En primer lugar el programa se inicializa mostrando en pantalla un mensaje de bienvenida en el que aparece el modelo del motor y la primera instrucción para iniciar el depliegue de los menus. Una vez en el despliegue de menús se muestran 2 opciones ($FlagSubMenu = 0$):

1. Ver datos

2. Ver alarmas

Si se selecciona la opción *Ver datos* mediante la pulsación del botón del encoder se desplegará el siguiente submenú con 7 opciones ($flagSubMenu = 1$):

1. Temperaturas
2. Presiones
3. Revoluciones
4. Niveles
5. Detector de niebla del cárter
6. Regulador de combustible
7. Salir

Cada una de las opciones de la lista anterior corresponden a variables del estado del motor (p. ej. temperatura). Mediante la pulsación del botón del encoder se muestra la lectura de los sensores instalados en las diferentes partes del motor, la siguiente imagen muestra un ejemplo para la opción de temperaturas:

Si se elige la opcion ver alarmas mediante la pulsación del botón del encoder se desplegará el siguiente submenu con 2 opciones ($flagSubMenu = 2$).

1. Alarmas
2. Salir

De igual modo que en el submenu ver datos, mediante la pulsación del botón del encoder se muestra las alarmas activas ($flagSubMenu = 4$).

3.4.2.2 Hardware de la interfaz de usuario

- Encoder
  La señal de movimiento entre los dierentes menús y submenús se realiza mediante un encoder provisto de un pulsador. Un encoder es un dispositivo que convierte el movimiento mecánico de un eje rotatorio en dos señales de pulsos digitales desfasada 90° . Dichas señales se leen mediante dos entradas digitales de la plataforma Arduino, el algoritmo de control en las líneas de código 154 a 162 (repetido para cada uno de los menús y submenús) evalúa el sentido en el que está rotando el eje [19] [26].

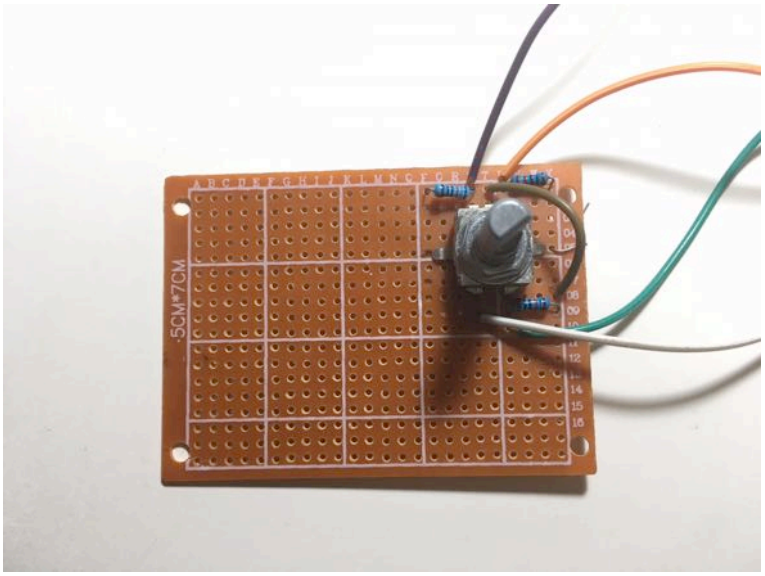

Figura 3.18: Montaje de encoder.

Fuente: Imagen de elaboración propia.

- Display LCD 20x4
  Para proporcionar información al usuario se usa un display LCD 20x4, es decir, veinte caracteres y cuatro líneas. La conexión con la plataforma Arduino se realiza mediante dos pines digitales y dos pines para alimentación (ver líneas de código de 164 a 705).

- Zumbador pasivo
  La señal acústica de alarma se realiza mediante un zumbador pasivo, se trata de un dispositivo que convierte una señal eléctrica en una onda de sonido. La alimentación del módulo se realiza conectando a los pines Vcc y GND de la plataforma Arduino y la entrada de señal a un pin digital (ver líneas de código 113, 750 y 755).

- Diodo emisor de luz *(Led)*
  La señales luminosas de se llevan a cabo mediante un *Light Emitting Diode* (LED) de color ámbar que indica los parámetros no estan en el rango de funcionamiento normal y se procede al arranque de bomba en standby, un LED de color rojo que indica alarma o parada del motor y además, un tercer un LED verde que indica el funcionamiento normal del motor.

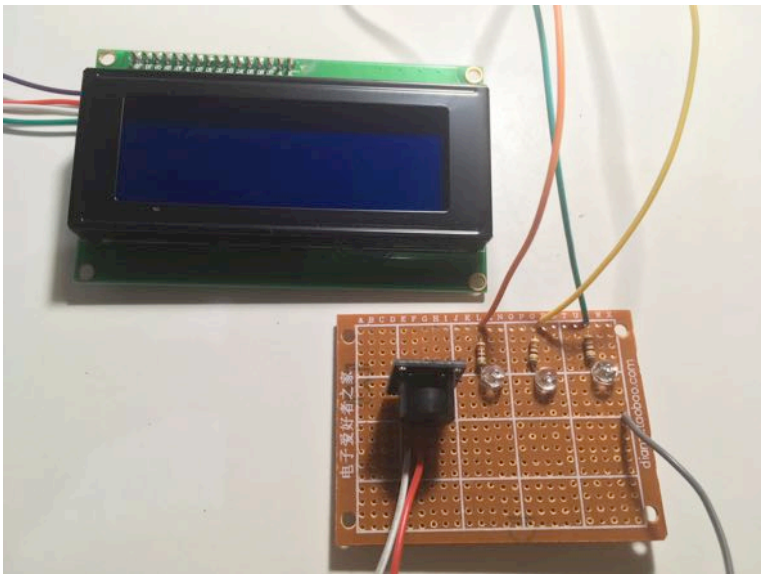

Figura 3.19: Montaje de elementos para interface de usuario.

Fuente: Imagen de elaboración propia.

4 Resultados

4.1 Datos obtenidos en la simulación

Se han realizado distintas simulaciones para comprobar el correcto funcionamiento del sistema así como la calibración de los sensores y de las alarmas para cumplir con las exigencias del fabricante del motor.

4.1.1 Sensores de temperatura

Para comprobar que la lectura es correcta se comprobó con un termómetro digital la temperatura indicada por el sensor. La siguiente tabla muestra los valores obtenidos:

Sensor termopar tipo K (°C)	**Sensor PT-100 (°C)**	**Termómetro digital (°C)**
18,90	17,50	18,00
25,30	25,00	25,20
60,20	60,00	60,40
84,80	85,00	84,80
100,20	100,50	100,30

Tabla 4.1: Verificación lectura de las temperaturas

4.1.2 Simulador de señal 4-20 mA del transductor de presión

Con el fin de verificar la linealidad entre la señal de voltaje de 1–5V proporcionada a la entrada del amplificador operacional y la salida de amperaje de 4-20 mA se instaló en paralelo a la salida del potenciómetro un voltímetro digital. Tras comprobar que el amplificador operacional disponía de un suministro de energía lo suficientemente alto para que 20 mA fluyan a través de la carga (*Rload*), se instaló en serie un amperímetro en la conexión de la resistencia (ver figura 4.1).

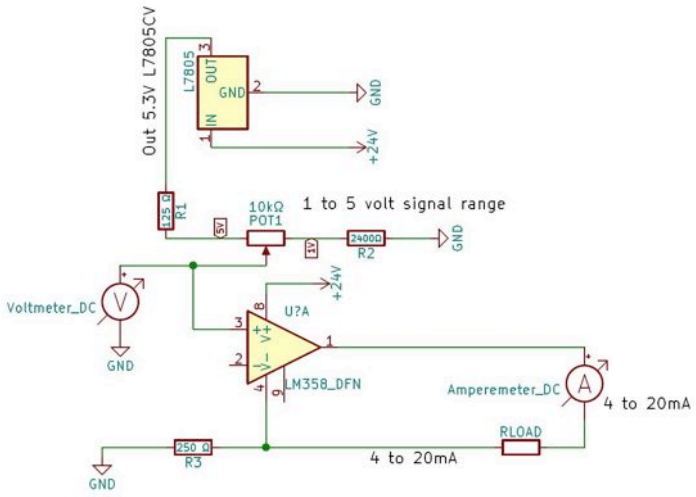

Figura 4.1: Puntos de medición en simulador señal 4-20 mA de transductores de presión.

Fuente: Imagen de elaboración propia.

Los datos obtenidos son los siguientes:

Voltaje (V)	**Intensidad de corriente (mA)**
1	4
2	8,5
3	12
4	16,5
5	20

4.2 Comprobación y errores en fase de programación

- Errores básicos de programación → Falta de llaves de cierre al final de las funciones, paréntesis mal colocados que provocan errores en la interpretación de comandos, falta de ; al final de funciones, etc.
- Errores en la interpretación de datos → Errores en las operaciones matemáticas para transformar los valores de voltaje a los valores de los parámetros.
- Errores en la estructura → Fallo en la estructura al mencionar los flagMenu para realizar la opción de salida.
- Error con la muestra de valores de los parámetros a traves del display LCD → Al imprimir los valores de temperatura el sensor actualizaba la información demasiado rápido y resultaba ilegible.

- Error en el cambio de menús → Se producían fallos al rotar el encoder que provocan fallos en el cambio entre menus y submenus.
- Errores con la señales luminosas → Cuando el sistema tenía una alarma permanecía el led verde encendido al mismo tiempo que el led rojo de alarma.
- Error en la medida de temperatura → A falta de disponer de una fuente de tensión estabilizada se produce un error $< 1\,\%$ respecto a la temperatura real.
- Error con el tiempo de pulsación → Se producía un error en el cambio de menús debido a la rapidez del programa para leer el estado del pulsador del encoder lo que hacía que seleccionara una opción diferente.
- Error al nombrar las opciones del menú → El movimiento del encoder produce cambios de posición a saltos de cuatro valores.
- Error a la hora de imprimir por pantalla los valores → Cuando se quería cambiar de opción las condiciones impuestas seguían cumpliéndose y no se realizaba el cambio a la lectura de otros parámetros.

4.3 Comprobación del correcto funcionamiento del prototipo

En esta sección se indican las pruebas realizadas con el fin de verificar el correcto funcionamiento del sistema. Para facilitar la búsqueda de fallos durante el desarollo del proyeto se han probado por separado los distintos sensores.

4.3.1 Inicialización del sistema

Al conectar la plataforma Arduino el sistema deberá en primer lugar cargar la pantalla de inicio mostrando en el display el mensaje de bienvenida, el programa deberá quedar a la espera de la instrucción del botón del encoder para entrar en el menu.

4.3.1.1 Procedimiento

- Realizar el montaje del circuito según el esquema eléctrico (figura A.5 del anexo A).

4.3.1.2 Resultados

Tras cargar el programa en la placa Arduino y conectarle el display y encoder, se comprueba el correcto encendido que muestra el mensaje de bienvenida así como los posteriores menús y submenús al interactuar con el encoder.

Figura 4.2: Mensaje inicial.

Fuente: Imagen de elaboración propia.

Figura 4.3: Menu principal.

Fuente: Imagen de elaboración propia.

4.3.2 Lectura de temperaturas

Cuando se selecciona la opcion de temperaturas que los sensores de temperatura obtienen los valores y los muestran mediante el display.

4.3.2.1 Procedimiento

Colocar los sensores en un recipiente con agua fría, otro con agua caliente y por último, fuera de los recipientes, midiendo la temperatura ambiente con un termómetro digital.

4.3.2.2 Resultados

Tras obtener las medidas de temperatura se observa una ligera desviación provocada por una fuente de alimentación poco precisa y el factor de error (0.2 %) del conversor de la termorresistemcia PT-100.

Figura 4.4: Conversor para termorresistencia PT-100.

Fuente: Imagen de elaboración propia.

4.3.3 Alarmas

Con los diferentes sensores se han realizado difrentes pruebas que simulan diferentes eventos de funcionamiento en el motor. Se comprueba que se enciende el LED rojo que indica alarma y se apaga el LED verde que indica el correcto funcionamiento del sistema.

4.3.3.1 Procedimiento

- Conectar, tal y como se indica en el esquema eléctrico (figura A.3 del anexo A) los presostatos, detectores de nivel y detector de niebla del cárter, además del transductor de presión y sensores de temperatura.
- Conectar la alimentación a la placa Arduino y realizar diferentes pruebas con los sensores de modo que representen eventos de importancia en el motor.

4.3.3.2 Resultados

Se comprueba que, cuando los parámetros están fuera de los valores normales de funcionamiento, se produce el estado de alarma, que activa el LED y la señal acústica. Al mismo tiempo, si se selecciona el menú alarmas aparece el código de la alarma activa.

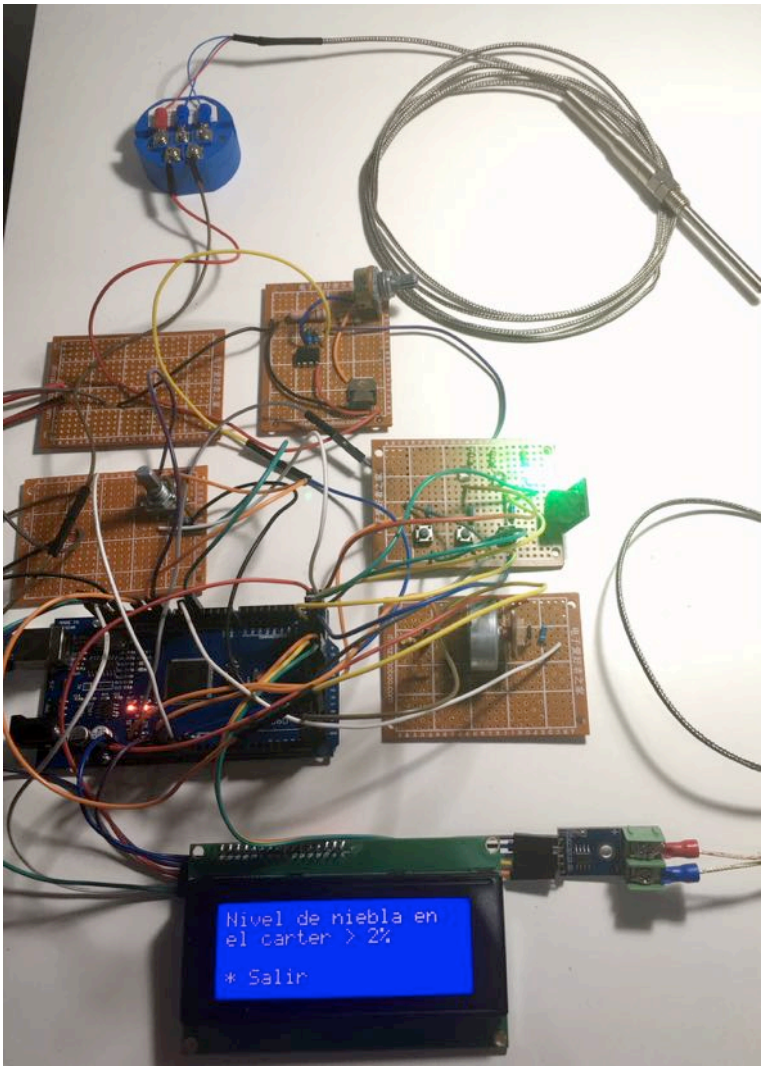

Figura 4.5: Prueba menús de alarma.

Fuente: Imagen de elaboración propia.

4.3.4 Lectura correcta del encoder rotatorio por parte de Arduino

Para comprobar que se leen los pulsos del encoder de manera correcta se ejecutaron una serie de instrucciones que permiten al programa que muestre mediante el monitor serie las posiciones del encoder, así como si se ha pulsado el botón.

4.3.4.1 Procedimiento

- Conectar y dar alimentacion a la placa Aduino únicamente con el encoder rotatorio.
- Ejecutar el programa en el que se ha añadido las instrucciones iniciar el monitor serie e imprimir por pantalla los valores de lectura del encoder.

- Abrir el monitor el monitor serie e interactuar con el encoder.

4.3.4.2 Resultados

Se comprueba que se produce la lectura de los pulsos del encoder. Para lograr un lectura lo más optimizada y definida posible fue necesario hacer ajustes para que los pulsos produzan valores múltiplos de 4.

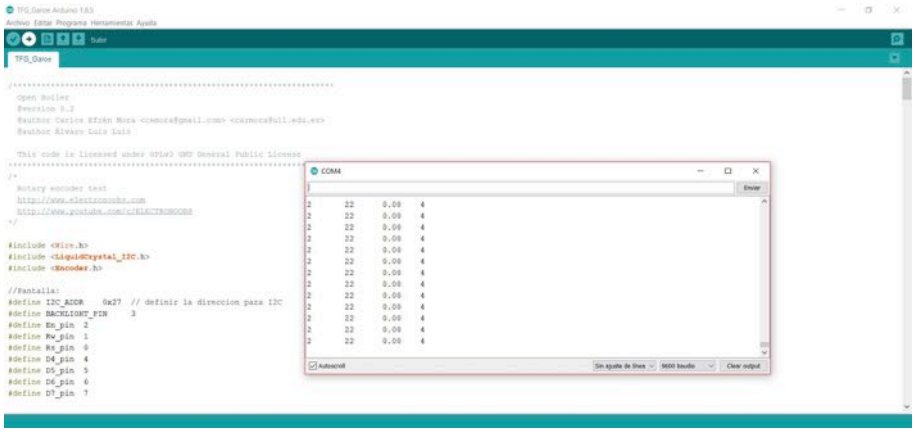

Figura 4.6: Monitor serie mostrando valores de encoder.

Fuente: Imagen de elaboración propia.

5 Discusión y conclusiones

Discusión y conclusión a partir de los resultados

Los incovenientes descubiertos en las plataformas electrónicas que controlan y monitorean el estado del motor principal MAK 12VM 32C, objeto de este trabajo, fueron el primordial motivo que impulsó crear un prototipo de electrónica que se presente como una alternativa al sistema existente.

Tras describir y observar los diferentes resutados obtenidos con la aplicación de la plataforma Arduino, se puede apreciar diferencias entre los datos arrojados por los sensores y la medida real de los mismos. Ello nos indica una menor precisión en la monitorización y posterior control del estado del motor.

Se puede observar en la variable de temperatura los cambios sucedidos. Concretamente, se observa diferencias del orden de $\pm 0,26^{\circ}$ C y $\pm 0,30^{\circ}$ C en el caso de los termopares tipo K y termorresistencias PT 100 respectivamente. Por otra parte, no hay evidencia de que los resultados de este estudio en la simulación de la señal de los transductores de presión, exista linealidad entre los valores de presión monitoreados en el motor y la señal de salida proporcionada por el simulador de la señal de los transductores.

Cabe a concluir en referencia a los sensores que disponen de una señal de salida todo/nada, que no existe ningún tipo de incompatibilidad en los sujetos experimentales a la hora de implementarlos en la plataforma Arduino.

Una vez analizados los resultados, se valora como factible una implementación real del sistema creado. Para su instalación no resulta necesario grandes modificaciones del sistema, basta con disponer las señales de salida de los sensores que se encuentran en las cajas de terminales sobre el motor (ver figuras 5.1, 5.2 y 5.3), así como el resto de elementos que componen la electrónica utilizada en este proyecto. Por otra parte, para garantizar la fiabilidad del sistema cabe la posibilidad de instalar un sistema en paralelo, para ello se tomará un punto de conexionado a la salida de los sensores del actual diseño. Este segundo sistema dispondrá de las mismas características que el principal incluido un equipo auxiliar para el control y la interface con el usuario.

Figura 5.1: Vista de las cajas de conexiones de los sensores del motor.

Fuente: Imagen de elaboración propia.

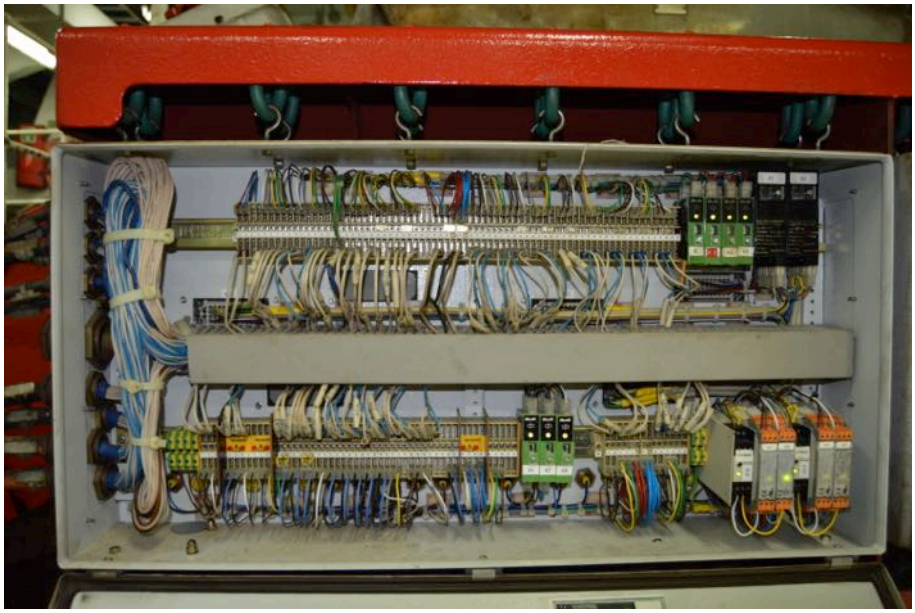

Figura 5.2: Vista de la caja de conexiones A (X1, X6, X8) de los sensores del motor.

Fuente: Imagen de elaboración propia.

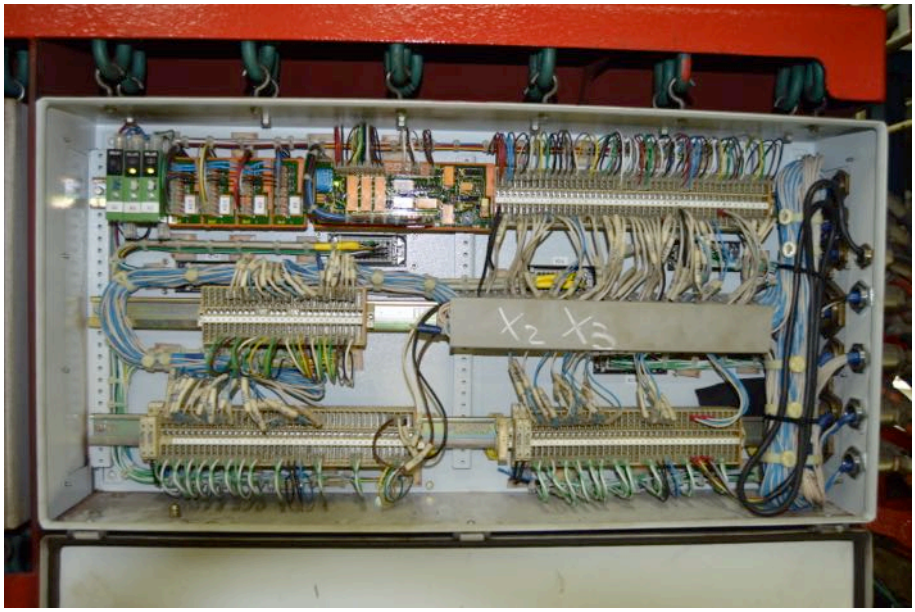

Figura 5.3: Vista de la caja de conexiones B (X2, X3) de los sensores del motor.

Fuente: Imagen de elaboración propia.

Posibles mejoras

La estructura del sistema permite añadir fácilmente sensores y sistemas por lo que también sirve como base para futuros proyectos. La siguiente lista muestra algunas mejores posibles que, por cuestiones de tiempo o presupuesto no se han podido incluir en este proyecto:

- Incluir sensores de tipo pick-up similares a los instalados en el motor.
- Mejorar la interfaz con el usuario a través de una pantalla TFT.
- Incluir los sensores transmisores de presión, presostatos y sondas de nivel.
- Incluir sensor detector de posicíon del regulador de combustible.
- Simplificar el diseño, realizar una placa PCB que reúna todos los componentes y conexiones y que a su vez se implemente con la placa Arduino.

Valoración personal

Tras recabar y analizar información sobre el tema y la posterior realización de este proyecto, he obtenido una serie de conclusiones:

El desarrollo y creación de estos problemas facilitan la adquisición de los conocimientos necesarios para su desarrollo, ya que permiten al usuario entender el origen del funcionamiento de los sistemas automatizados, los cuales ven incrementado su protagonismo día

a día, ya que facilitan tanto la recogida información como la realización de actividades secundarias en sistemas, ahorrando tiempo al personal para el desarrollo de labores más importantes.

Además, es un método alternativo de estudio ya que, al ser un proyecto desarrollado a partir de los conocimientos adquiridos durante los años de estudios, permite a los estudiantes tanto refrescar conocimientos, debido a la necesidad de disponer de una buena base teórica.

Cabe agregar que, la elaboración de este proyecto me ha permitido profundizar en los conocimientos adquiridos durante la carrera académica. Por ejemplo, la realización de diagramas de flujo, una buena organización y/o planificación; esto es, disponiendo de una buena teórica, se podrán abordar diferentes problemáticas que surjan a lo largo de la carrera profesional.

Finalmente espero que esta aportación sirva como base para futuros proyectos de investigación para mejorar la monitorización de los parámetros de funcionamiento y el control de motores. Por esta razón aliento de manera especial a los alumnos del grado en tecnologías marinas a mejorar el sistema creado para de este trabajo de fin de grado.

Bibliografía

[1] AUTOMATION, S., "Detector de niebla de aceite en el cárter (Oil Mist Detector)". [Página web], 2018 [consultado 26 de agosto de 2018]. Http://rubedate.e.telefonica.net/schaller-2.htm.

[2] Barloworld Finanzauto., *Control y Supervisión*. Service pro ed., 2010. Formación de servicios, Carlos Rico.

[3] Caterpillar Inc., *Diesel Engine Control and Monitoring System*. Sensors ed., 2008. Training Center Kiel.

[4] CONTAVAL, "¿Porque hay sondas PT-100 que tienen 3 hilos?" [Página web], 2017 [consultado 12 de junio de 2018]. Http://www.contaval.es/sondas-pt-100-tienen-3-hilos/.

[5] CORPORATION, K., "¿Qué es un sensor de proximidad inductivos?" [Página web], 2018 [consultado 26 de marzo de 2018]. Https://www.keyence.com.mx/ss/products/sensor/sensorbasics/proximity/info/.

[6] DE REGULACION Y CONTROL SRC, S., "¿QUÉ ES UN SENSOR PT100?" [Página web], 2017 [consultado 10 de junio de 2018]. Http://srcsl.com/que-es-un-sensor-pt100/.

[7] EINGENEERING, O., "Sensor de nivel". [Página web], 2018 [consultado 15 de julio de 2018]. Https://es.omega.com/prodinfo/sondas-de-nivel-medicion.html.

[8] EINGENEERING, O., "Termopar: Tipos y Aplicaciones". [Página web], 2018 [consultado 24 de junio de 2018]. Https://cl.omega.com/prodinfo/termopar.html.

[9] EINGENEERING, O., "Transductor de presión". [Página web], 2018 [consultado 6 de junio de 2018]. Https://es.omega.com/prodinfo/transductores-de-presion.html.

[10] ELECTRONICS, L., "Voltage-to-current signal conversion". [Página web], 2018 [consultado 13 de junio de 2018]. Http://www.learningelectronics.net/vol3/chpt8/7.html.

[11] FACTORY, A. G., "Midiendo temperatura MAX6675". [Página web], 2019 [consultado 20 de mayo de 2018]. Https://www.geekfactory.mx/tutoriales/tutoriales-pic/midiendo-temperatura-max6675/.

[12] García, R., "Oil Mist Detector-Detector de Niebla en Cárter". [Página web], 2017 [consultado 27 de agosto de 2018]. Https://ingenieromarino.com/oil-mist-detector-detector-de-niebla-en-carter/.

[13] Garma Electrónica, S., "Medidores/detectores de nivel capacitivos: para detección y medición de nivel en líquidos". [Página web], 2018 [consultado 19 de julio de 2018]. Http://www.interempresas.net/Mantenimiento/FeriaVirtual/Producto-Medidores-detectores-de-nivel-capacitivos-142449.html.

[14] Industry, D., "SENSOR DE VELOCIDAD DE ROTACIÓN / MAGNÉTICO / PICK-UP". [Página web], 2018 [consultado 3 de julio de 2018]. Http://www.directindustry.es/prod/frank-w-murphy-ltd/product-12515-548019.html.

[15] Interface, S. S. P., "SPI - Serial Peripheral Interface". [Página web], 2006 [consultado 21 de mayo de 2018]. Http://www.mct.net/faq/spi.html.

[16] Keen, D., "¿Cómo funciona un relé SPDT?" [Página web], 2018 [consultado 22 de agosto de 2018]. Https://www.geniolandia.com/13121672/como-funciona-un-rele-spdt.

[17] Ltda., I. M., "¿QUÉ ES ARDUINO?" [Página web], 2018 [consultado 4 de abril de 2018]. Http://arduino.cl/que-es-arduino/.

[18] Mauricio, J., "Sensor Pick Up". [Página web], 2018 [consultado 4 de julio de 2018]. Https://es.scribd.com/document/235455217/Sensor-Pick-Up.

[19] Mecafenix, F., "Encoder ¿como funciona? y sus tipos". [Página web], 2017 [consultado 11 de julio de 2018]. Http://www.ingmecafenix.com/automatizacion/encoder/.

[20] Meganeboy, D., "Sensores en el automóvil". [Página web], 2018 [consultado 9 de julio de 2018]. Http://www.aficionadosalamecanica.net/sensores2-modelos.htm.

[21] Miyachi, A., "K Type Thermocouple [Termopar tipo K]". [Página web], 2018 [consultado 17 de junio de 2018]. Http://spanish.amadamiyachi.com/glossary/glossktypethermocouple.

[22] motion, C., "Cómo funcionan los sensores inductivos". [Página web], 2018 [consultado 26 de agosto de 2018]. Https://www.zettlex.com/es/articles/sensores-inductivos-funcionan/.

[23] Motorlan, "Sondas térmicas en motores eléctricos: PTC, NTC, RTD... ¿Qué las diferencia?" [Página web], 2017 [consultado 13 de junio de 2018]. Http://www.motorlan.es/es/sondas-termicas-motores-electricos-ptc-ntc-rtd-las-diferencia/.

[24] Nolla, X., "¿Cómo funciona un transmisor de presión?" [Página web], 2017 [consultado 7 de junio de 2018]. Https://www.bloginstrumentacion.com/productos/como-funciona-un-transmisor-de-presion/.

[25] Playground, A., "Estructura de un programa". [Página web], 2017 [consultado 20 de abril de 2018]. Https://playground.arduino.cc/ArduinoNotebookTraduccion/Structure.

[26] products company, E., "¿Qué es un Encoder?" [Página web], 2017 [consultado 13 de julio de 2018]. Http://encoder.com/blog/encoder-basics/que-es-un-encoder/.

[27] Rowe, D., *CRC Handbook of thermoelectrics*. CRC Press; 1 edition, 1995, ISBN 978-0849301469.

[28] Sicoms, "Sistemas de Monitoreo de Concentración de neblina y temperatura de salpicadura en el aceite". [Página web], 2018 [consultado 26 de agosto de 2018]. Http://www.momacsa.com/images/pdf/motcomSiCOMS130405spanish.pdf.

[29] srl, A., "CONTROLES DE NIVEL". [Página web], 2017 [consultado 16 de julio de 2018]. Http://www.aecosensors.com/index.cfm/fuseaction=prodottiCatPadre/id=2/aeco-level-controls/languageID=ES.

[30] TR3SDLAND, "Componentes – Sensor de temperatura NTC". [Página web], 2011 [consultado 16 de junio de 2018]. Https://www.tr3sdland.com/2011/12/componentes-el-sensor-ntc/.

[31] Turmero, P., "Dispositivos electrónicos Caterpillar". [Página web], 2018 [consultado 7 de julio de 2018]. Https://www.monografias.com/trabajos104/dispositivos-electronicos-caterpillar/dispositivos-electronicos-caterpillar.shtml.

[32] Veloso, C., "FUNCIONAMIENTO DE UN SENSOR DE TEMPERATURA". [Página web], 2016 [consultado 8 de junio de 2018]. Http://www.electrontools.com/Home/WP/2016/03/30/funcionamiento-de-un-sensor-de-temperatura/.

[33] Veloso, C., "QUE ES Y PARA QUE SIRVE ARDUINO". [Página web], 2017 [consultado 15 de abril de 2018]. Http://www.electrontools.com/Home/WP/2016/04/20/que-es-y-para-que-sirve-arduino/.

[34] Ventura, V., "MAX6675. Conversor analógico-digital para sondas de termopar K con compensación de unión fría y comunicaciones SPI". [Página web], 2016 [consultado 23 de mayo de 2018]. Https://polaridad.es/max6675-termopar-k-thermocouple-temperatura-compensacion-union-fria-spi-arduino/.

Anexos

A Esquemas eléctricos

En este apartado se presentan los planos realizados eb el presente proyecto y que a contiuación se enumeran.

- Plano Nº1: Simulador señal 4-20mA
- Plano Nº2: Sensores de temperatura
- Plano Nº3: Sensores simulados a través de pulsadores
- Plano Nº4: Simulador de velocidad del motor y sensores pick-up
- Plano Nº5: Elementos de comunicación del usuario con la placa Arduino.

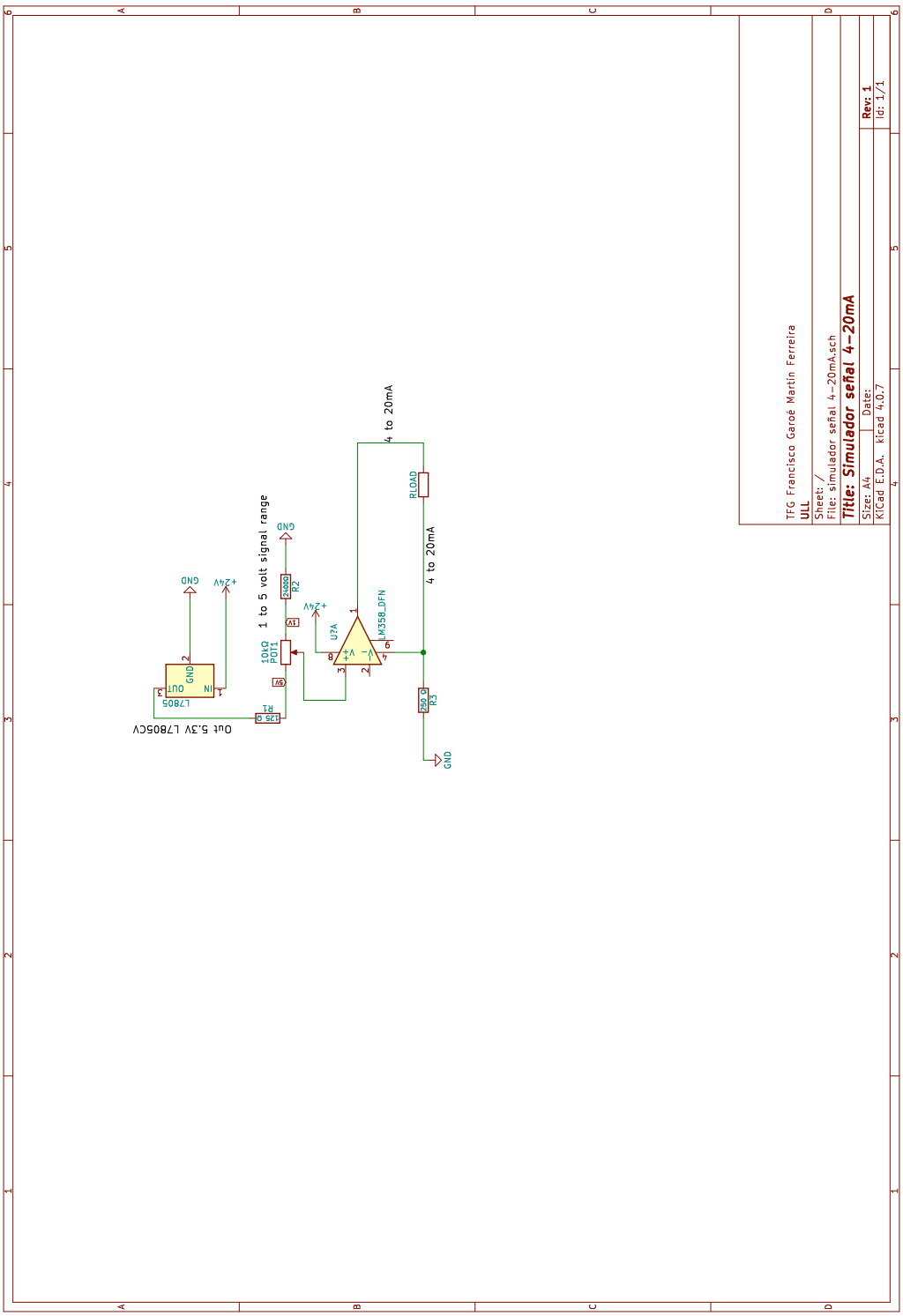

Figura A.1: Simulador señal 4-20 mA

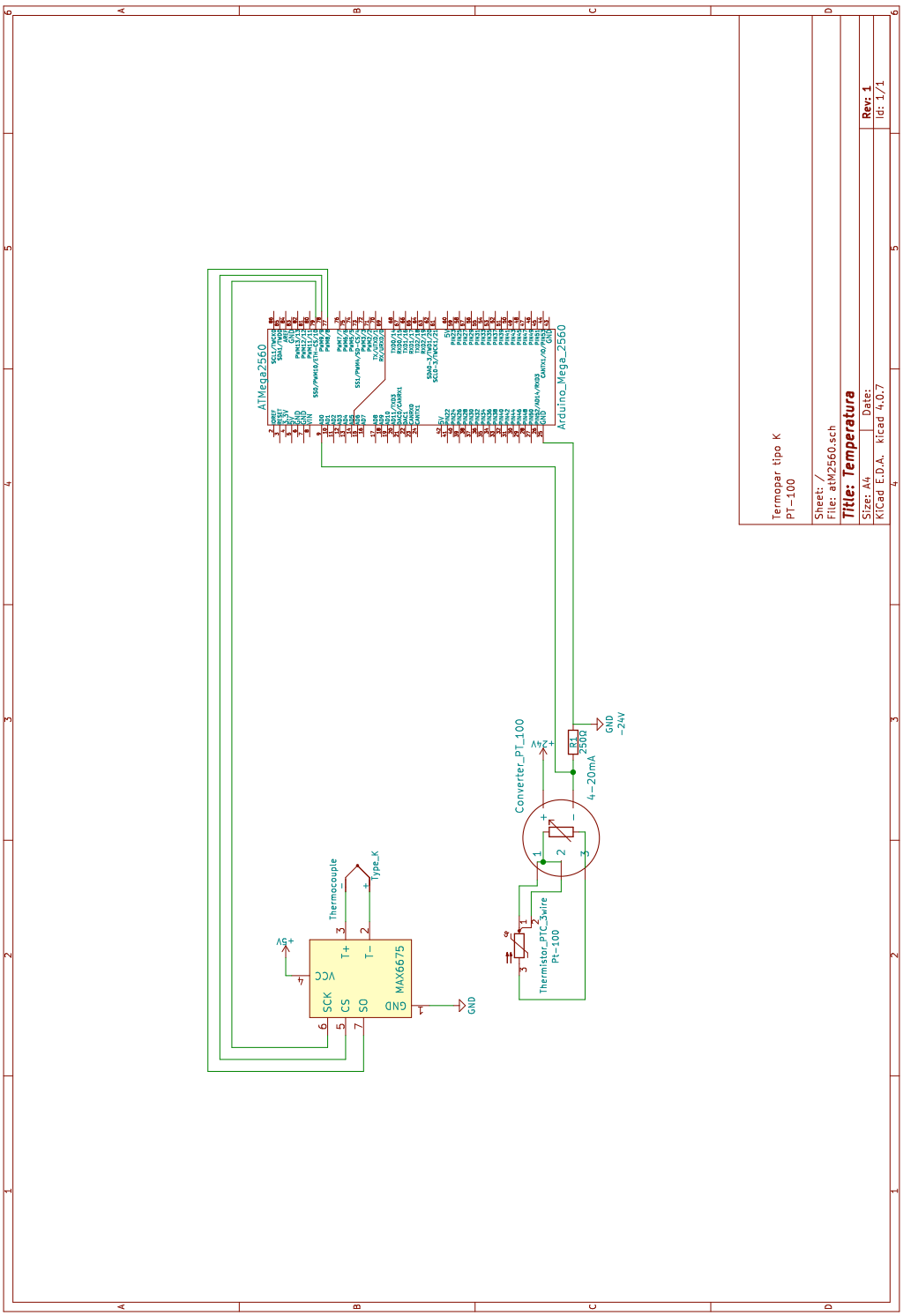

Figura A.2: Sensores de temperatura PT-100 y termopar tipo K

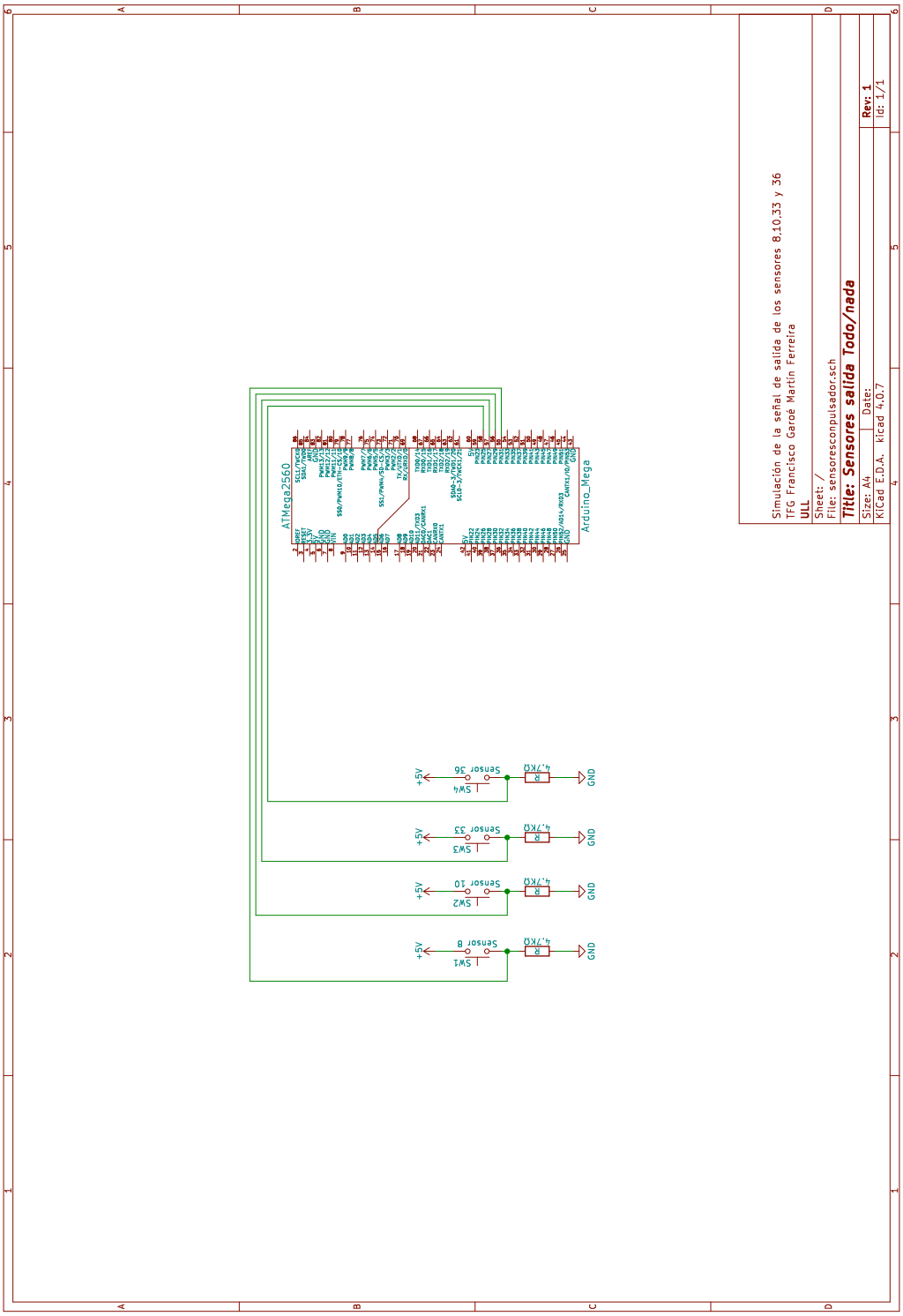

Figura A.3: Sensores todo/nada: sonda de nivel, presostatos y detector de niebla en el cárter

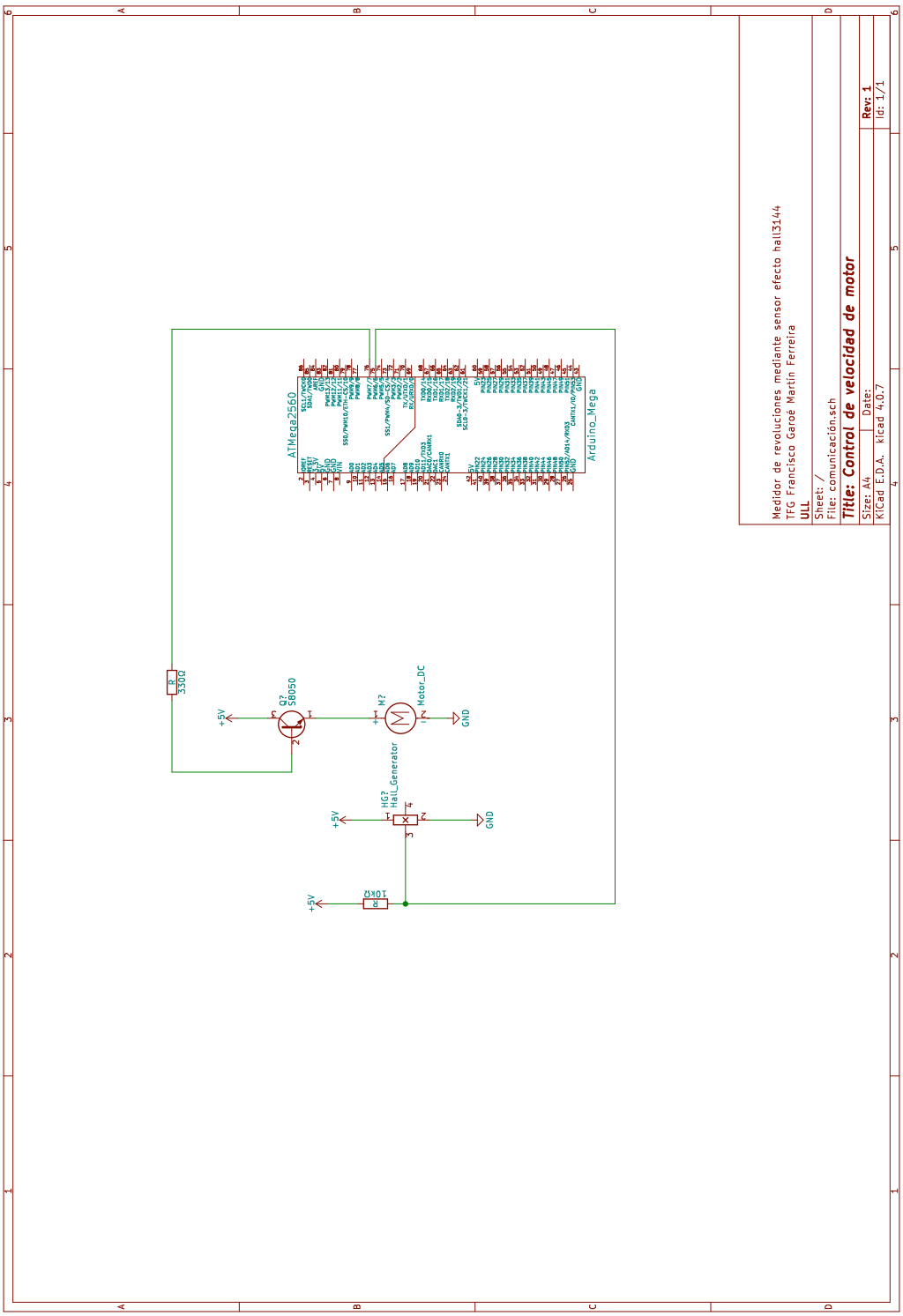

Figura A.4: Simulación de velocidad del motor y medidor de revoluciones

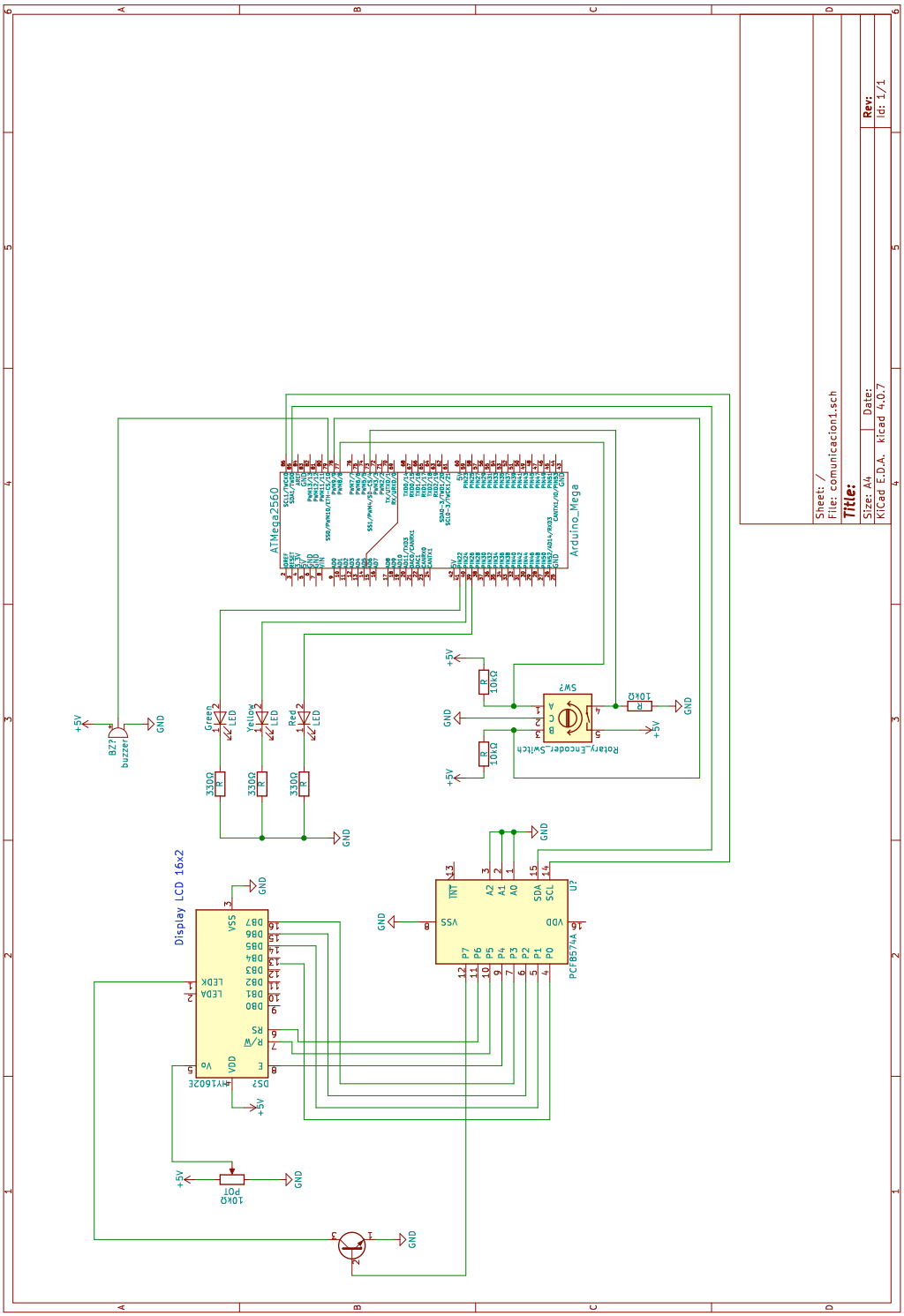

Figura A.5: Interface con usuario mediante encoder rotatorio display LCD, señales luminosas y señal sonora

B Código fuente

```
1  /*
2    *Prototipo orientado a su ejecución en un ardware libre para el
        monitoreo y control de parámetros de funcionamiento de un motor de
        combustión interna marino.
3    *@author Francisco Garoé Martín Ferreira <cgaroemf1@gmail.com> <
        alu0100834000@ull.edu.es>
4    *@author Carlos Efrén Mora Luis <carmora@ull.edu.es>
5  */
6
7  /*
8    Parte del código fuente para la gestión del encoder se ha tomado de:
9    Rotary encoder test
10   http://www.electronoobs.com
11   http://www.youtube.com/c/ELECTRONOOBS
12 */
13
14 #include <Wire.h>
15 #include <LiquidCrystal_I2C.h>
16 #include <Encoder.h>
17
18 //Pantalla:
19 #define I2C_ADDR    0x27  // definir la direccion para I2C
20 #define BACKLIGHT_PIN     3
21 #define En_pin  2
22 #define Rw_pin  1
23 #define Rs_pin  0
24 #define D4_pin  4
25 #define D5_pin  5
26 #define D6_pin  6
27 #define D7_pin  7
28
29 //Menu Configuration
30 #define maxMainMenuPos 4.0   //Main menu positions
31 #define maxDataMenuPos 24.0   //Data submenu positions
32 #define maxAlarmMenuPos 2.0  //Alarms submenu position
33 int menuOption = 0;          //Main menu option
34 int flagMenu = 0;            //Selected menu option (Main menu start
       position)
35 int flagSubMenu = 0;         //Selectected submenu
```

```
36
37 //Menu encoder configuration
38 #define clk 8
39 #define data 9
40 float counter = 0.0;
41 int State;
42 int LastState;
43
44 int anterior = 0;
45 int estado = 0;
46 int entrar = 0;
47
48 //Inputs
49 byte menuButton = 4;
50
51 // Teropar type K
52 #include <max6675.h>
53 int ktcSO = 33;
54 int ktcCS = 34;
55 int ktcCLK = 35;
56 float t2;
57 MAX6675 ktc(ktcCLK, ktcCS, ktcSO);
58
59 // Variables globales:
60 byte arrowDown[8] = { 0B00111, 0B00101, 0B00111, 0B00000, 0B00000, 0
      B00000, 0B00000, 0B00000}; // Caracteres personalizados para el LCD
61
62 // PT100:
63 #define tinput A0
64 float tvoltage;
65 float temp;
66 float t1;
67
68 /* PULSADORES NO */
69 const int Bpresion = 25;
70 const int Bnivel = 26;
71 const int Bniebla = 27;
72 int Vpresion;
73 int Vnivel;
74 int Vniebla;
75
76 /* 4-20mA */
77 const int sensorPin = A1;    // seleccionar la entrada para el sensor
78 int sensorValue;          // variable que almacena el valor (0 a 1023)
79 float value;             // variable que almacena el voltaje (0.0 a 5.0)
80 float pressure;
81
82 /* motor */
83 const int hallSensorPin = 5;                          // pin donde esta
      conectado el sensor hall 5
```

```
84  const unsigned long sampleTime = 1000;
85
86
87  //LEDS Y SONIDO
88  int ledG = 22;
89  int ledR = 23;
90  int ledY = 24;
91  int Sound = 28;
92  long oldPosition  = -999;
93
94  Encoder myEnc(8, 9);
95
96  // Se instancia el LCD con los pines indicados:
97  LiquidCrystal_I2C  lcd(I2C_ADDR, En_pin, Rw_pin, Rs_pin, D4_pin, D5_pin,
        D6_pin, D7_pin);
98
99  void setup() {
100   //LCD start
101   lcd.begin(20, 4);
102   lcd.setBacklightPin(BACKLIGHT_PIN, POSITIVE);
103   lcd.setBacklight(HIGH);
104   lcd.createChar(1, arrowDown);
105
106   //Inputs
107   pinMode(menuButton, INPUT);
108   pinMode (clk, INPUT);
109   pinMode (data, INPUT);
110   pinMode( ledG, OUTPUT);
111   pinMode( ledR, OUTPUT);
112   pinMode(motor,  OUTPUT);
113   pinMode( Sound, OUTPUT);
114
115
116   //Welcome message
117   do {
118     lcd.setCursor(4, 0);
119     lcd.print("MAK 12VM 32C");
120     lcd.setCursor(0, 2);
121     lcd.print("Para acceder al menu");
122     lcd.setCursor(0,3);
123     lcd.print("      Pulse OK");
124   } while (!digitalRead(menuButton));
125
126   // Reads the initial state of the clock pin
127   LastState = digitalRead(clk);
128   delay(300);
129   lcd.clear();
130
131   Serial.begin(9600);
132   LastState = digitalRead(clk);
```

```
133   //  delayMicroseconds(3000);
134 }
135
136 void loop() {
137   menuDisplay();
138   readInputs();
139   //  printLcd();
140   alarmActions();
141 }
142
143 void menuDisplay() {
144   long newPosition;
145  /* Serial.print(flagSubMenu);
146   Serial.print("\t");
147   Serial.print(flagMenu);
148   Serial.print("\t");
149   Serial.print(counter);
150   Serial.print("\t");
151   Serial.print(menuOption);
152   Serial.print("\n");
153   */
154   if (flagSubMenu == 0) {
155     newPosition = myEnc.read();
156     delay(5);
157     if (newPosition != oldPosition) {
158       oldPosition = newPosition;
159     }
160     if (newPosition >= maxMainMenuPos) myEnc.write(maxMainMenuPos);
161     if (newPosition <= 0) myEnc.write(0);
162     menuOption = newPosition;
163
164     lcd.setCursor(0, 0);
165
166     switch (menuOption) {
167       case 0:
168         lcd.setCursor(0, 0);
169         lcd.print("* Ver datos     ");
170         lcd.setCursor(0, 1);
171         lcd.print("  Ver alarmas   ");
172         flagMenu = 1;
173         break;
174       case 4:
175         lcd.setCursor(0, 0);
176         lcd.print("  Ver datos     ");
177         lcd.setCursor(0, 1);
178         lcd.print("* Ver alarmas   ");
179         flagMenu = 2;
180         break;
181     }
182   }
```

```
183
184
185   if (digitalRead(menuButton) && flagMenu == 1) {
186     flagSubMenu = 1;
187     lcd.clear();
188     myEnc.write(0);
189     delay(150);
190   }
191   if (digitalRead(menuButton) && flagMenu == 2) {
192     flagSubMenu = 2;
193     lcd.clear();
194     myEnc.write(0);
195     delay(150);
196   }
197   if (digitalRead(menuButton) && (flagMenu == 17 || flagMenu == 22)) {
198     lcd.clear();
199     flagSubMenu = 0;
200     myEnc.write(0);
201     delay(150);
202   }
203
204
205
206   if (flagSubMenu == 1) {
207     newPosition = myEnc.read();
208     delay(5);
209     if (newPosition != oldPosition) {
210       oldPosition = newPosition;
211     }
212     if (newPosition >= maxDataMenuPos) myEnc.write(maxDataMenuPos);
213     if (newPosition <= 0 ) myEnc.write(0);
214     menuOption = newPosition;
215     lcd.setCursor(0, 0);
216
217     switch (menuOption) {
218       case 0:
219         lcd.setCursor(0, 0);
220         lcd.print("* Temperaturas      ");
221         lcd.setCursor(0, 1);
222         lcd.print("  Presiones         ");
223         lcd.setCursor(0, 2);
224         lcd.print("  Revoluciones      ");
225         lcd.setCursor(0, 3);
226         lcd.print("  Niveles           ");
227         flagMenu = 11;
228         break;
229       case 4:
230         lcd.setCursor(0, 0);
231         lcd.print("  Temperaturas      ");
232         lcd.setCursor(0, 1);
```

```
233         lcd.print("* Presiones          ");
234         lcd.setCursor(0, 2);
235         lcd.print("  Revoluciones       ");
236         lcd.setCursor(0, 3);
237         lcd.print("  Niveles            ");
238         flagMenu = 12;
239         break;
240       case 8:
241         lcd.setCursor(0, 0);
242         lcd.print("  Temperaturas       ");
243         lcd.setCursor(0, 1);
244         lcd.print("  Presiones          ");
245         lcd.setCursor(0, 2);
246         lcd.print("* Revoluciones       ");
247         lcd.setCursor(0, 3);
248         lcd.print("  Niveles            ");
249         flagMenu = 13;
250         break;
251       case 12:
252         lcd.setCursor(0, 0);
253         lcd.print("  Temperaturas       ");
254         lcd.setCursor(0, 1);
255         lcd.print("  Presiones          ");
256         lcd.setCursor(0, 2);
257         lcd.print("  Revoluciones       ");
258         lcd.setCursor(0, 3);
259         lcd.print("* Niveles            ");
260         flagMenu = 14;
261         break;
262       case 16:
263         lcd.setCursor(0, 0);
264         lcd.print("  Presiones          ");
265         lcd.setCursor(0, 1);
266         lcd.print("  Revoluciones       ");
267         lcd.setCursor(0, 2);
268         lcd.print("  Niveles            ");
269         lcd.setCursor(0, 3);
270         lcd.print("* Detector niebla    ");
271         flagMenu = 15;
272         break;
273       case 20:
274         lcd.setCursor(0, 0);
275         lcd.print("  Revoluciones       ");
276         lcd.setCursor(0, 1);
277         lcd.print("  Niveles            ");
278         lcd.setCursor(0, 2);
279         lcd.print("  Detector niebla    ");
280         lcd.setCursor(0, 3);
281         lcd.print("* Reg combustible    ");
282         flagMenu = 16;
```

```
283         break;
284       case 24:
285         lcd.setCursor(0, 0);
286         lcd.print("  Niveles           ");
287         lcd.setCursor(0, 1);
288         lcd.print("  Detector niebla   ");
289         lcd.setCursor(0, 2);
290         lcd.print("  Reg combustible   ");
291         lcd.setCursor(0, 3);
292         lcd.print("* Salir             ");
293         flagMenu = 17;
294         break;
295     }
296   }
297
298   if (flagSubMenu == 2) {
299     newPosition = myEnc.read();
300     delay(5);
301     if (newPosition != oldPosition) {
302       oldPosition = newPosition;
303     }
304     if (newPosition >= maxDataMenuPos) myEnc.write(maxDataMenuPos);
305     if (newPosition <= 0 ) myEnc.write(0);
306     menuOption = newPosition;
307     lcd.setCursor(0, 0);
308
309     switch (menuOption) {
310       case 0:
311         lcd.setCursor(0, 0);
312         lcd.print("* Alarmas  ");
313         lcd.setCursor(0, 1);
314         lcd.print("  Salir     ");
315         flagMenu = 21;
316         break;
317       case 4:
318         lcd.setCursor(0, 0);
319         lcd.print("  Alarmas  ");
320         lcd.setCursor(0, 1);
321         lcd.print("* Salir     ");
322         flagMenu = 22;
323         break;
324     }
325   }
326   //*****************************************************************
       ALARMAS
327   if (digitalRead(menuButton) && flagMenu == 21) {
328     flagSubMenu = 4;
329     lcd.clear();
330     myEnc.write(0);
331     delay(150);
```

```
332   }
333   if (flagSubMenu == 4) {
334     newPosition = myEnc.read();
335     delay(5);
336     if (newPosition != oldPosition) {
337       oldPosition = newPosition;
338     }
339     if (newPosition >= maxDataMenuPos) myEnc.write(maxDataMenuPos);
340     if (newPosition <= 0 ) myEnc.write(0);
341     menuOption = newPosition;
342     lcd.setCursor(0, 0);
343
344     switch (menuOption) {
345       case 0:
346         if ((t1) > 70) {
347           lcd.setCursor(0, 0);
348           lcd.print("Alta temperatura de ");
349           lcd.setCursor(0, 1);
350           lcd.print("aceite a la entrada ");
351           lcd.setCursor(0, 2);
352           lcd.print("del motor "); lcd.print(t1); lcd.write(byte(1));
                lcd.print("C  ");
353           lcd.setCursor(0, 3);
354           lcd.print("* Salir             ");
355           flagMenu = 41;
356         }
357         if ((t2) > 490) {
358           lcd.setCursor(0, 0);
359           lcd.print("Alta temperatura de ");
360           lcd.setCursor(0, 1);
361           lcd.print("salida gases de      ");
362           lcd.setCursor(0, 2);
363           lcd.print("escape 1A   "); lcd.print(t1); lcd.write(byte(1));
                lcd.print("C");
364           lcd.setCursor(0, 3);
365           lcd.print("* Salir             ");
366           flagMenu = 41;
367         }
368         if ((pressure) <= 100) {
369           lcd.setCursor(0, 0);
370           lcd.print("Baja presion de Fuel");
371           lcd.setCursor(0, 1);
372           lcd.print("Oil a la entrada del");
373           lcd.setCursor(0, 2);
374           lcd.print("motor "); lcd.print(pressure); lcd.print(" kPa    "
                );
375           lcd.setCursor(0, 3);
376           lcd.print("* Salir             ");
377           flagMenu = 41;
378         }
```

```
379         if (Vpresion == HIGH) {
380           lcd.setCursor(0, 0);
381           lcd.print("Baja presion de agua");
382           lcd.setCursor(0, 1);
383           lcd.print("HT a la entrada     ");
384           lcd.setCursor(0, 2);
385           lcd.print("del motor < 210 kPa ");
386           lcd.setCursor(0, 3);
387           lcd.print("* Salir             ");
388           flagMenu = 41;
389         }
390         if (Vnivel == HIGH)
391         {
392           lcd.setCursor(0, 0);
393           lcd.print("Nivel de aceite en  ");
394           lcd.setCursor(0, 1);
395           lcd.print("la bandeja del motor");
396           lcd.setCursor(0, 2);
397           lcd.print("Alto                ");
398           lcd.setCursor(0, 3);
399           lcd.print("* Salir             ");
400           flagMenu = 41;
401         }
402         if (Vniebla == HIGH)
403         {
404           lcd.setCursor(0, 0);
405           lcd.print("Nivel de niebla en  ");
406           lcd.setCursor(0, 1);
407           lcd.print("el carter > 2%      ");
408           lcd.setCursor(0, 3);
409           lcd.print("* Salir             ");
410           flagMenu = 41;
411         }
412       }
413     }
414     if (digitalRead(menuButton) && flagMenu == 41) {
415       lcd.clear();
416       flagSubMenu = 2;
417       myEnc.write(0);
418       delay(150);
419     }
420     //------------------------------------------------------------------ if
          (digitalRead(menuButton) && flagMenu == 11) {
421       flagSubMenu = 3;
422       lcd.clear();
423       myEnc.write(0);
424       delay(150);
425     }
426     if (flagSubMenu == 3) {
427       newPosition = myEnc.read();
```

```
428     delay(5);
429     if (newPosition != oldPosition) {
430       oldPosition = newPosition;
431     }
432     if (newPosition >= maxDataMenuPos) myEnc.write(maxDataMenuPos);
433     if (newPosition <= 0 ) myEnc.write(0);
434     menuOption = newPosition;
435     lcd.setCursor(0, 0);
436
437     switch (menuOption) {
438       case 0:
439         lcd.setCursor(0, 0);
440         lcd.print("Lube oil "); lcd.print(t1); lcd.write(byte(1)); lcd.
              print("C   ");
441         lcd.setCursor(0, 1);
442         lcd.print("Temp cyl 1A "); lcd.print(t2); lcd.write(byte(1));
              lcd.print("C");
443         lcd.setCursor(0, 2);
444         lcd.print("Salir");
445         flagMenu = 31;
446         break;
447       case 4:
448         lcd.setCursor(0, 0);
449         lcd.print("Lube oil "); lcd.print(t1); lcd.write(byte(1)); lcd.
              print("C");
450         lcd.setCursor(0, 1);
451         lcd.print("Temp cyl 1A "); lcd.print(ktc.readCelsius()); lcd.
              write(byte(1)); lcd.print("C");
452         lcd.setCursor(0, 2);
453         lcd.print("* Salir");
454         flagMenu = 32;
455     }
456   }
457   if (digitalRead(menuButton) && flagMenu == 32) {
458     lcd.clear();
459     flagSubMenu = 1;
460     myEnc.write(0);
461     delay(150);
462   }
463
464   //-------------------------------------------------------------
        VARIABLES DE PRESION
465   if (digitalRead(menuButton) && flagMenu == 12) {
466     flagSubMenu = 5;
467     lcd.clear();
468     myEnc.write(0);
469     delay(150);
470   }
471   if (flagSubMenu == 5) {
472     newPosition = myEnc.read();
```

```
473     delay(5);
474     if (newPosition != oldPosition) {
475       oldPosition = newPosition;
476     }
477     if (newPosition >= maxDataMenuPos) myEnc.write(maxDataMenuPos);
478     if (newPosition <= 0 ) myEnc.write(0);
479     menuOption = newPosition;
480     lcd.setCursor(0, 0);
481     switch (menuOption) {
482       case 0:
483         lcd.setCursor(0, 1);
484         lcd.print("P. de FO "); lcd.print(pressure, 1); lcd.print(" kPa
                "");
485         if (Vpresion == LOW)
486         {
487           lcd.setCursor(0, 0);
488           lcd.print("P de agua HT Ok     ");
489         }
490         if (Vpresion == HIGH)
491         {
492           lcd.setCursor(0, 0);
493           lcd.print("P de agua HT Alta   ");
494         }
495         lcd.setCursor(0, 3);
496         lcd.print("Salir               ");
497         flagMenu = 33;
498         break;
499
500       case 4:
501         if (Vpresion == LOW)
502         {
503           lcd.setCursor(0, 0);
504           lcd.print("P de agua HT Ok     ");
505         }
506         if (Vpresion == HIGH)
507         {
508           lcd.setCursor(0, 0);
509           lcd.print("P de agua HT Alta   ");
510         }
511         lcd.setCursor(0, 3);
512         lcd.print("* Salir             ");
513         flagMenu = 34;
514         break;
515     }
516   }
517   if (digitalRead(menuButton) && flagMenu == 34) {
518     lcd.clear();
519     flagSubMenu = 1;
520     myEnc.write(0);
521     delay(150);
```

```
522   }
523    /*-----------------------------------------------------------------
          REVOLUCIONES */
524   int rpm;
525   if (digitalRead(menuButton) && flagMenu == 13) {
526     flagSubMenu = 6;
527     lcd.clear();
528     myEnc.write(0);
529     delay(150);
530   }
531   if (flagSubMenu == 6) {
532     newPosition = myEnc.read();
533     delay(5);
534     if (newPosition != oldPosition) {
535       oldPosition = newPosition;
536     }
537     if (newPosition >= maxDataMenuPos) myEnc.write(maxDataMenuPos);
538     if (newPosition <= 0 ) myEnc.write(0);
539     menuOption = newPosition;
540     lcd.setCursor(0, 0);
541     switch (menuOption) {
542       case 0:
543         lcd.setCursor(0, 0);
544         lcd.print("Rpm =  "); lcd.print(rpm);
545         lcd.setCursor(0, 3);
546         lcd.print("Salir               ");
547         flagMenu = 39;
548         break;
549
550       case 4:
551         lcd.setCursor(0, 0);
552         lcd.print("Rpm =  "); lcd.print(rpm);
553         lcd.setCursor(0, 3);
554         lcd.print("* Salir             ");
555         flagMenu = 40;
556         break;
557     }
558   }
559   if (digitalRead(menuButton) && flagMenu == 40) {
560     lcd.clear();
561     flagSubMenu = 1;
562     myEnc.write(0);
563     delay(150);
564   }
565    /*-----------------------------------------------------------------
          VARIABLES DE NNIVEL */
566   if (digitalRead(menuButton) && flagMenu == 14) {
567     flagSubMenu = 7;
568     lcd.clear();
569     myEnc.write(0);
```

```
570       delay(150);
571     }
572     if (flagSubMenu == 7) {
573       newPosition = myEnc.read();
574       delay(5);
575       if (newPosition != oldPosition) {
576         oldPosition = newPosition;
577       }
578       if (newPosition >= maxDataMenuPos) myEnc.write(maxDataMenuPos);
579       if (newPosition <= 0 ) myEnc.write(0);
580       menuOption = newPosition;
581       lcd.setCursor(0, 0);
582       switch (menuOption) {
583         case 0:
584           if (Vnivel == LOW)
585           {
586             lcd.setCursor(0, 0);
587             lcd.print("Nivel de aceite en  ");
588             lcd.setCursor(0, 1);
589             lcd.print("la bandeja del motor");
590             lcd.setCursor(0, 2);
591             lcd.print("Ok                  ");
592           }
593           if (Vnivel == HIGH)
594           {
595             lcd.setCursor(0, 0);
596             lcd.print("Nivel de aceite en  ");
597             lcd.setCursor(0, 1);
598             lcd.print("la bandeja del motor");
599             lcd.setCursor(0, 2);
600             lcd.print("Alto                ");
601           }
602           lcd.setCursor(0, 3);
603           lcd.print("Salir               ");
604           flagMenu = 35;
605           break;
606 
607         case 4:
608           if (Vnivel == LOW)
609           {
610             lcd.setCursor(0, 0);
611             lcd.print("Nivel de aceite en  ");
612             lcd.setCursor(0, 1);
613             lcd.print("la bandeja del motor");
614             lcd.setCursor(0, 2);
615             lcd.print("Ok                  ");
616           }
617           if (Vnivel == HIGH)
618           {
619             lcd.setCursor(0, 0);
```

```
620           lcd.print("Nivel de aceite en  ");
621           lcd.setCursor(0, 1);
622           lcd.print("la bandeja del motor");
623           lcd.setCursor(0, 2);
624           lcd.print("Alto                ");
625         }
626         lcd.setCursor(0, 3);
627         lcd.print("* Salir             ");
628         flagMenu = 36;
629         break;
630     }
631   }
632   if (digitalRead(menuButton) && flagMenu == 36) {
633     lcd.clear();
634     flagSubMenu = 1;
635     myEnc.write(0);
636     delay(150);
637   }
638    /*
       ----------------------------------------------------------------------------
        NIEBLA */
639   if (digitalRead(menuButton) && flagMenu == 15) {
640     flagSubMenu = 8;
641     lcd.clear();
642     myEnc.write(0);
643     delay(150);
644   }
645   if (flagSubMenu == 8) {
646     newPosition = myEnc.read();
647     delay(5);
648     if (newPosition != oldPosition) {
649       oldPosition = newPosition;
650     }
651     if (newPosition >= maxDataMenuPos) myEnc.write(maxDataMenuPos);
652     if (newPosition <= 0 ) myEnc.write(0);
653     menuOption = newPosition;
654     lcd.setCursor(0, 0);
655     switch (menuOption) {
656       case 0:
657         if (Vniebla == LOW)
658         {
659           lcd.setCursor(0, 0);
660           lcd.print("Nivel de niebla en  ");
661           lcd.setCursor(0, 1);
662           lcd.print("el carter < 2%      ");
663         }
664         if (Vniebla == HIGH)
665         {
666           lcd.setCursor(0, 0);
667           lcd.print("Nivel de niebla en  ");
```

```
668             lcd.setCursor(0, 1);
669             lcd.print("el carter > 2%      ");
670           }
671           lcd.setCursor(0, 3);
672           lcd.print("Salir               ");
673           flagMenu = 37;
674           break;
675
676         case 4:
677           if (Vniebla == LOW)
678           {
679             lcd.setCursor(0, 0);
680             lcd.print("Nivel de niebla en  ");
681             lcd.setCursor(0, 1);
682             lcd.print("el carter < 2%      ");
683           }
684           if (Vniebla == HIGH)
685           {
686             lcd.setCursor(0, 0);
687             lcd.print("Nivel de niebla en  ");
688             lcd.setCursor(0, 1);
689             lcd.print("el carter > 2%      ");
690           }
691           lcd.setCursor(0, 3);
692           lcd.print("* Salir             ");
693           flagMenu = 38;
694           break;
695       }
696     }
697     if (digitalRead(menuButton) && flagMenu == 38) {
698       lcd.clear();
699       flagSubMenu = 1;
700       myEnc.write(0);
701       delay(150);
702     }
703
704 }
705 //
      ----------------------------------------------------------------------------------------

706  //PT-100
707    tvoltage = analogRead(tinput);
708    if (tvoltage > 204) {
709      tvoltage = map(tvoltage, 205, 1023, 0, 2000);
710      temp = tvoltage;
711      t1 = temp / 10.0;
712      t1 = round (t1);
713    }
714    //TERMOPAR
715    t2 = (ktc.readCelsius());
```

```
716
717   // PRESIONES
718   Vpresion = digitalRead(Bpresion);
719
720   //NIVEL
721   Vnivel = digitalRead(Bnivel);
722
723   //NIEBLA
724   Vniebla = digitalRead(Bniebla);
725
726
727   sensorValue = analogRead(sensorPin);
728   value = fmap(sensorValue, 0, 1023, 0.1, 5.0);
729   if (value <= 1.1) {
730     pressure = 100;
731   }
732   else {
733     pressure = value * 200.0;
734     pressure = round(pressure);
735   }
736 }
737
738 // cambio de escala entre floats
739 float fmap(float x, float in_min, float in_max, float out_min, float
      out_max)
740 {
741   return (x - in_min) * (out_max - out_min) / (in_max - in_min) +
        out_min;
742 }
743
744 void alarmActions() {
745
746   /* Red led and alarm sound */
747   if (((t1) > 70) || (Vpresion == HIGH) || (Vnivel == HIGH) || (Vniebla
        == HIGH) || ((pressure) < 280)) {
748     digitalWrite (ledR, HIGH);
749     analogWrite(motor, 0);
750     tone(buzzer, 500);
751   }
752   else {
753     digitalWrite (ledR, LOW);
754     analogWrite(motor, 100);
755     //  noTone(buzzer);
756   }
757   /* Yellow led */
758   if ((pressure) < 280) {
759     digitalWrite (ledY, HIGH);
760   }
761   else {
762     digitalWrite (ledY, LOW);
```

```
763   }
764   /* Green led */
765   if ((t1) < 65 && (Vpresion == LOW) && (Vnivel == LOW) && (Vniebla ==
          LOW) && ((pressure) >= 350)) {
766     digitalWrite (ledG, HIGH);
767     analogWrite(motor, 40);
768   }
769   else digitalWrite (ledG, LOW);
770 }
771 int getRPM() /*codigo para obtener revoluciones*/
772 {
773   int count = 0;
774   boolean countFlag = LOW;
775   unsigned long currentTime = 0;
776   unsigned long startTime = millis();
777   while (currentTime <= sampleTime)
778   {
779     if (digitalRead(hallSensorPin) == HIGH)
780     {
781       countFlag = HIGH;
782     }
783     if (digitalRead(hallSensorPin) == LOW && countFlag == HIGH)
784     {
785       count++;
786       countFlag = LOW;
787     }
788     currentTime = millis() - startTime;
789   }
790   int countRpm = int(60000 / float(sampleTime)) * count;
791   return countRpm;
792 }
```