ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO

SISTEMA DE SENSORIZACIÓN INALÁMBRICA PARA VEHÍCULOS DE COMPETICIÓN

TRABAJO FINAL DE GRADO

Realizado por: ALBERTO TOBARRA TABOADA

Dirigido por: RAFAEL MASOT PÉRIS

Departamento de Ingeniería Electrónica

22/09/2014

Documento 1

MEMORIA

ÍNDICE:

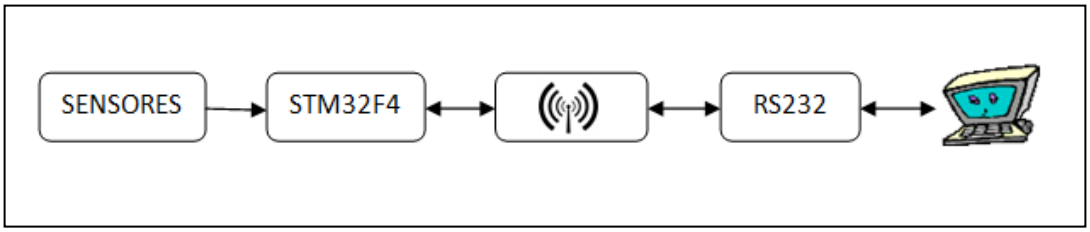
1. OBJETO 5
2. JUSTIFICACIÓN DEL PROYECTO 5
2.1 Antecedentes 5
2.2 Justificación del proyecto 6
3. FACTORES A CONSIDERAR 6
3.1 Normativa 6
3.2 Licencias 6
3.3 Condiciones del encargo 7
4. SOLUCIONES ALTERNATIVAS 8
4.1 Soluciones alternativas 8
4.1.1 Sistema de comunicación 8
4.1.2 Medidas sensadas 8
4.2 Criterios de selección 8
4.3 Justificación de la solución adoptada 9
4.3.1 Selección del sistema de comunicación 9
4.3.2 Selección del sistema de sensado 9
5. DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN ADOPTADA 10
5.1 Sistema y subsistemas 10
5.1.1 Subsistema eléctrico 10
5.1.1.1 Circuito de alimentación 10
5.1.1.2 Conexionado eléctrico 11
5.1.2 Subsistema electrónico 13
5.1.2.1 Procesador 13
5.1.2.2 Componentes de sensado 14
5.1.2.3 Hardware de comunicación 20
6. JUSTIFICACIÓN DE LA SOLUCIÓN ADOPTADA 21
6.1 Subsistema Eléctrico 21
6.1.1 Fuente de alimentación (Batería) 21

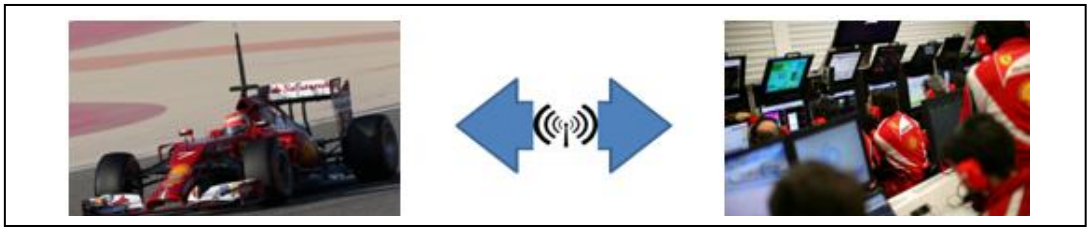
6.1.2 Cableado de conexión de pines 21
6.1.3 Convertidor Serie RS232 22
6.1.4 Cable transmisión de datos 22
6.2 Subsistema Electrónico 23
6.2.1 Procesador 23
6.2.2 Componentes de sensado 24
6.2.3 Regulador de tensión LM7805 24
6.2.4 Hardware de radiofrecuencia 25
7. SOFTWARE 25
7.1 Programación del micro controlador 25
7.1.1 Flujogramas 26
7.1.2 Main.c 27
7.1.3 BoardConfig.c 31
7.1.4 BoardConfig.h 34
7.2 Programación Interfaz de Usuario 34
7.2.1 Interfaz de Usuario 39

MEMORIA

1. OBJETO

La presente memoria se refiere al diseño de un sistema de sensorización inalámbrica para un vehículo de competición y visualización gráfica de las diferentes señales recibidas. El proyecto incluye un diseño de programación, un diseño eléctrico, así como el manejo de componentes electrónicos y las propias tareas de instalación y acondicionamiento.





2. JUSTIFICACIÓN DEL PROYECTO

2.1 ANTECEDENTES

Debido al proceso anterior de sensorización con sistemas de hardware anticuados y poco precisos, y a consecuencia de querer modernizarse a la vez que mejora la calidad y rapidez del sistema, se ha realizado un estudio de viabilidad de un nuevo sistema de sensorización que incorpora una nueva programación para su utilización en un nuevo soporte de hardware.

2.2 JUSTIFICACIÓN DEL PROYECTO

El proyecto es realizado en el seno del departamento de ingeniería electrónica a petición del equipo de competición Fórmula Student de la Universidad Politécnica de Valencia, que desea un sistema de telemetría que proporcione rapidez, calidad y seguridad del producto permitiendo a los Ingenieros de carrera interpretar los datos recogidos durante una prueba o carrera y utilizarlos para sintonizar correctamente el coche y obtener así un rendimiento óptimo de sus vehículos.

3. FACTORES A CONSIDERAR

3.1 NORMATIVA

En el deporte del automovilismo, los últimos desarrollos incluyen telemetría bidireccional que permite a los ingenieros actualizar calibraciones en el coche en tiempo real. En la Fórmula 1, la telemetría bidireccional surgió en la década de 1990 y consistió en una pantalla de mensajes del panel de control que el equipo podría actualizar. Su desarrollo continuó hasta mayo de 2001, cuando se permitió por primera vez en los coches. En 2002, los equipos fueron capaces de cambiar los mapas del motor y desactivar los sensores del motor mientras el coche estaba en la pista. Para la temporada 2003, la FIA prohibió la telemetría bidireccional de la Fórmula 1, sin embargo, la tecnología se puede utilizar en otros tipos de carreras o en los coches de carretera.

3.2 LICENCIAS

Se obtendrá la licencia de software correspondiente al sistema de interfaz de usuario mediante el programa Matlab Guide.

Para la lectura de la recepción de los datos serie correspondientes a las medidas de los diferentes sensores analógicos (potenciómetros), se necesitará también la licencia de software del Isis Proteus.

En cuanto al CooCox, programa utilizado para la implementación de la programación, no se requerirá de licencia puesto que es de software gratuito.

3.3 CONDICIONES DEL ENCARGO

Se implementará un sistema de transmisión de datos entre un vehículo en movimiento y una estación fija, en la cual se podrá ver graficados los valores recibidos, así como grabar datos en tiempo real del vehículo en cuestión a cierta distancia. Esto permitirá a los Ingenieros de datos detectar cualquier anomalía en el funcionamiento del vehículo.

El sistema deberá cumplir con las siguientes premisas de funcionamiento:

- La medida, envío y lectura de toda la secuencia de datos se realizará cada segundo.
- El tipo de datos a sensar serán procedentes de tres sensores de señal analógica.
- La transmisión de datos deberá ser efectiva a larga distancia.
- El proceso deberá ser controlado perfectamente y de manera intuitiva por el usuario desde la estación fija.
- Para el control del sistema se requerirá de una interfaz gráfica de usuario.
- Los datos podrán ser almacenados en diferentes documentos de texto para su posterior estudio.

El soporte hardware emplazado en el vehículo estará constituido por un micro controlador Stm32f4 Discóvery de 32 bits basado en la arquitectura Harvard con núcleo ARM Cortex-M4, el cual nos permitirá un procesado más rápido, debido a que el procesador puede acceder a memoria por dos caminos diferentes, uno para datos y otro para programa.

En cuanto a la transmisión de datos, contaremos con un módulo HM-TRP (transmisión por radio frecuencia) con el que se comunicará con la estación fija a través de una comunicación serie a larga distancia. En la estación fija se dispondrá de un receptor conectado a un convertidor RS232 a través del cual se codificarán los datos enviados y recibidos para su correcta lectura.

El proceso será controlado a través del interfaz de usuario creado con Matlab GUIDE. Éste dispondrá de un selector de puertos COM (6) y de un selector de velocidades (3). Dependiendo del puerto que detecte el dispositivo fijo se seleccionará uno u otro y la velocidad se configurará por programa para el funcionamiento a 9600. Una vez configurado los puertos y la velocidad se dispone de tres botones. Uno de ellos configurado con función toggle de marcha/paro del sistema; otro botón pulsador para guardar los datos, que serán almacenados en un documento de texto.txt y en una hoja de datos Excel para así poder graficar o realizar los estudios pertinentes; y finalmente

un botón para salir de la aplicación, el cuál al ser pulsador nos preguntará si realmente estamos seguros de la acción de salir.

4. SOLUCIONES ALTERNATIVAS

4.1 SOLUCIONES ALTERNATIVAS

Para la ejecución del proyecto se han contemplado los siguientes planteamientos:

4.1.1 SISTEMA DE COMUNICACIÓN

- Implementar el sistema de comunicación serie inalámbrica a través del protocolo SPI.
- Implementar el sistema de comunicación serie inalámbrica a través del protocolo USART

4.1.2 MEDIDAS SENSADAS

- Adquisición de un sensor de temperatura.
- Adquisición de un sensor de presión.
- Adquisición de un sensor de revoluciones.
- Utilización de potenciómetros.

4.2 CRITERIOS DE SELECCIÓN

El criterio de selección de los diferentes planteamientos priorizará los sistemas que permitan un nivel de adquisición de datos y calidad mayor, siempre y cuando se mantenga el compromiso entre calidad-precio y se encuentre al alcance del usuario.

4.3 JUSTIFICACIÓN DE LA SOLUCIÓN ADOPTADA

4.3.1 SELECCIÓN DEL SISTEMA DE COMUNICACIÓN

Se empleará la opción de comunicación serie tipo UART. El Receptor Asíncrono Universal / transmisor (UART) realiza bytes de datos y transmite los bits individuales de una manera secuencial. En el destino, un segundo UART re-ensambla los bits en bytes completos. Cada UART contiene un registro de desplazamiento , que es el método fundamental de la conversión entre las formas de serie y paralelos. Esta opción de transmisión en serie de información digital (bits) a través de un solo cable u otro medio (en nuestro caso, estándar RS232), es menos costosa que la transmisión en paralelo a través de múltiples cables.

4.3.2 SELECCIÓN DEL SISTEMA DE SENSADO

La solución adoptada para la generación de señales, debido al coste elevado y la dificultad de la obtención de los diferentes sensores, es la utilización de potenciómetros. Un potenciómetro es un resistor cuyo valor de resistencia es variable. De esta manera, al conectarlo en serie se puede sensar la diferencia de potencial y así generar una señal analógica variable en el rango de tensión de alimentación.

5. DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN ADOPTADA

5.1 SISTEMA Y SUBSISTEMAS

El sistema está compuesto por dos subsistemas formados según el ámbito de aplicación, esto es, subsistema eléctrico y electrónico.

5.1.1 SUBSISTEMA ELÉCTRICO

5.1.1.1 CIRCUITO DE ALIMENTACIÓN

El equipo constará de una fuente de alimentación en nuestro caso procedente de la batería del vehículo (12V), a través de la cual se obtendrá un nivel de tensión continua de 5V utilizando un LM7805 con la que poder alimentar el circuito electrónico correspondiente a la Discovery Stm32F4.

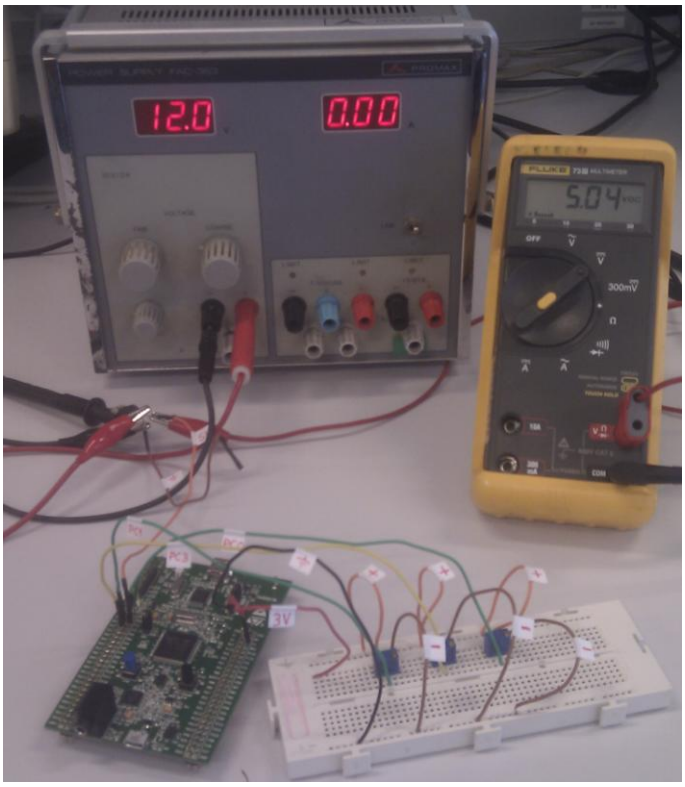


IMAGEN1: Circuito de alimentación

5.1.1.2 CONEXIONADO ELÉCTRICO

El sistema de cableado interconexionará tanto los elementos de comunicación por RF como el circuito de sensado con la placa electrónica, la cual contiene el micro controlador y los diferentes pines. El cableado será de diversos tipos:

En cuanto a la conexión de los pines correspondientes a la comunicación, se utilizará cables Dupont flexibles de 2.54mm redondos o planos, según disponibilidad, con uno de los extremos con jumper tipo hembra de crimp.

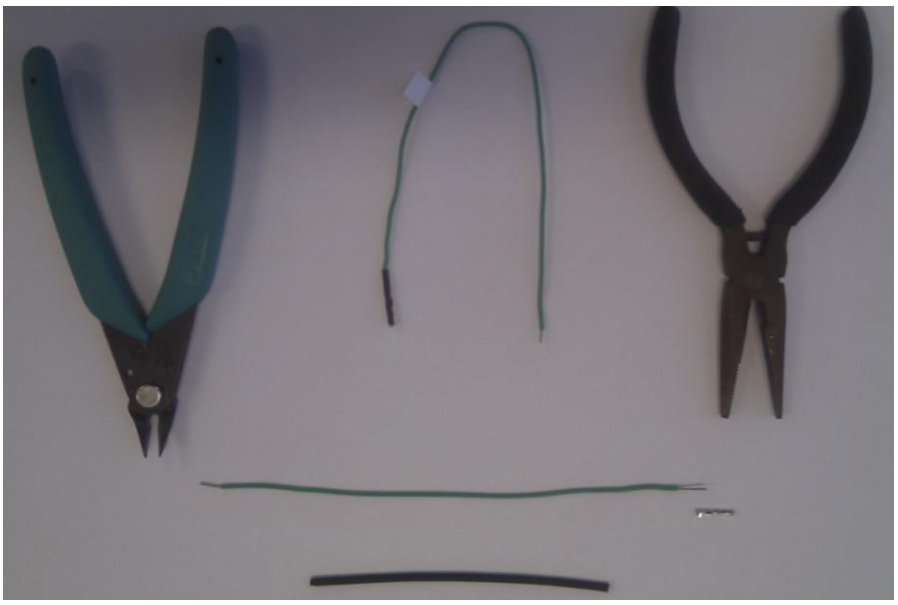


IMAGEN 2: Cables Dupont flexibles 2.54mm

Para la conexión de la estación fija con el emisor/receptor de RF se empleará un cable apantallado con un terminal USB que se conectará a la estación fija (PC) y, un conector tipo RS-232 que se conectará con el transmisor de RF. El conector USB nos proporciona a través del PC una alimentación de 3.3V, lo necesario para alimentar al transmisor, y el RS-232 (Recommended Standard 232, también conocido como EIA/TIA RS-232C) es una interfaz que designa una norma para el intercambio de una serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE (Equipo de Comunicación de datos) que consiste en una conector tipo DB-9 macho (de 9 pines), una versión más económica que el DB-25 e incluso más extendida para ciertos periféricos.

IMAGEN 3: RS 232 DB9

5.1.2 SUBSISTEMA ELECTRÓNICO

5.1.2.1 PROCESADOR

El soporte hardware del cual se requerirá será de un Kit de placa electrónica constituida por un micro controlador STM32F407VGT6 con Núcleo ARM Cortex-M4F 32-bit, 1 MB Flash, 192 KB de RAM en un paquete LQFP100, el cual nos permitirá un procesado más rápido, debido a que el procesador puede acceder a memoria por dos caminos diferentes, uno para datos y otro para programa. Además, la misma placa electrónica incorpora diversos pines dedicados al ADC, se puede utilizar como fuente de alimentación de 3V y 5V, y también incorpora entre muchas cosas más 6 salidas para la comunicación serie USART, de las cuales 2 de ellas de alta velocidad y por tanto una de ellas empleada para el envío de los datos.

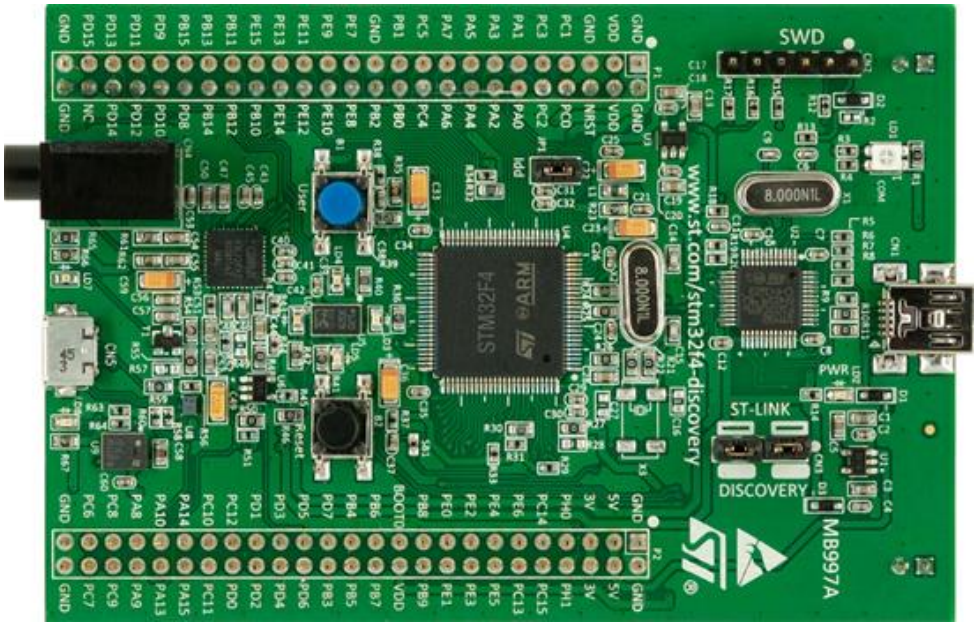


IMAGEN 4: Discovery stm32f4

5.1.2.2 COMPONENTES DE SENSADO

Para la generación de las señales se han seleccionado 3 sensores de un vehículo que proporcionan medidas analógicas. Estos son el sensor de posición del acelerador, el sensor del caudal del aire y el sensor de temperatura del motor.

Los sensores citados se detallan a continuación:

- **SENSOR DE POSICIÓN DEL ACELERADOR (TPS)**

El sensor de posición del acelerador (TPS) es un potenciómetro (un tipo de resistor variable) con una amplia variedad de modelos. La computadora suministra voltaje y tierra al sensor. El sensor tiene una pieza de tipo rotativo o de tipo lineal y si está montado en el motor la pieza viene acoplada al acelerador de manera que se mueven juntos. El sensor envía una señal de voltaje a la computadora indicando la posición del acelerador y la señal se incrementa cuando se abre el acelerador.

La computadora usa la posición del acelerador para determinar el estado de operación neutro (acelerador cerrado), crucero (parcialmente en neutro) o aceleración intensa (acelerador muy abierto) y entonces puede controlar adecuadamente las mezclas de aire-combustible, avance del encendido, velocidad en neutro, etc.

A continuación se describe el sensor de posición del acelerador.

Descripción del sensor TPS:

El sensor TPS es un potenciómetro rotatorio que le envía a la computadora una señal, la cual indica en qué posición se encuentra la mariposa de aceleración.

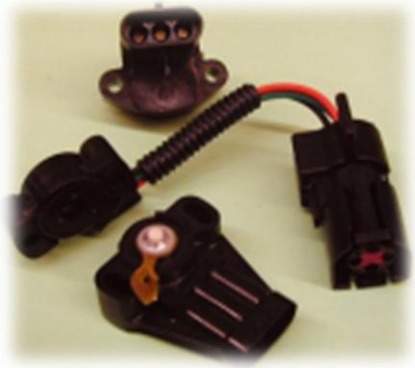


IMAGEN 5: Sensor TPS

El sensor TPS cuenta con un conector de 3 terminales, las cuales son:

• 5V

• Señal

• Tierra

Nota:

1) Normalmente las terminales de los extremos son el voltaje de alimentación y la tierra; y la terminal del centro es la señal de referencia.

2) El voltaje de alimentación del sensor es por lo general de 5V para cualquier marca.

IMAGEN 6: Conector sensor TPS

Localización típica:

El sensor TPS generalmente se encuentra montado en el exterior del armazón del acelerador y conectado al eje del acelerador.

IMAGEN 7: Ubicación del sensor TPS

- **SENSOR CAUDAL DEL AIRE**

Se localiza entre el filtro de aire y la válvula de mariposa y mide la masa de aire que entra al múltiple de admisión. Existen dos tipos: el de hilo y el de película caliente. Ambos sensores operan bajo el siguiente principio: un delgado alambre o una delgada película de platino que se encuentra en el medidor de masa de aire acompañada de un sensor de temperatura que es calentada mediante electricidad. La unidad de mando recibe una señal eléctrica que es proporcional a la masa de aire que entra al múltiple, con este dato la unidad de mando calcula la cantidad necesaria de combustible.

IMAGEN 8: Medidor de masa de aire

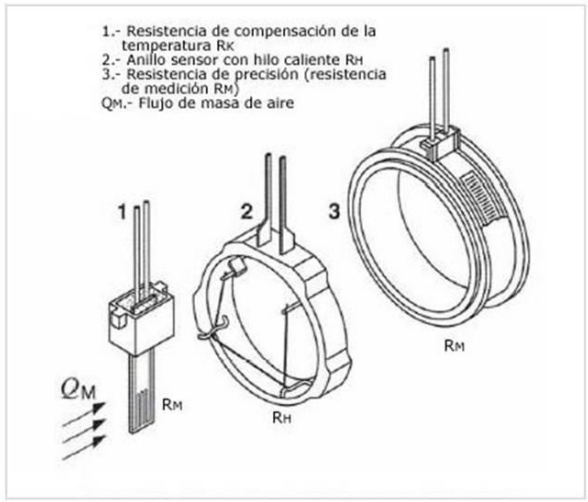


IMAGEN 9: Despiece completo del medidor de masa de aire por hilo caliente

- **SENSOR TEMPERATURA DEL MOTOR**

Su objetivo es conocer la temperatura de motor a partir de la temperatura del líquido refrigerante del mismo, informando a la unidad de control para que regule la mezcla y el momento de encendido del combustible (campo de medición - 40...+130 °C).

IMAGEN 10: Sensores de temperatura NTC para líquidos

Estructura y funcionamiento

Dentro de un cuerpo hay montada una resistencia termo sensible de medición, de material semiconductor. Normalmente tiene ella un coeficiente de temperatura negativo (NTC), raramente un coeficiente de temperatura positivo (PTC), es decir, que su resistencia disminuye o aumenta drásticamente al subir la temperatura.

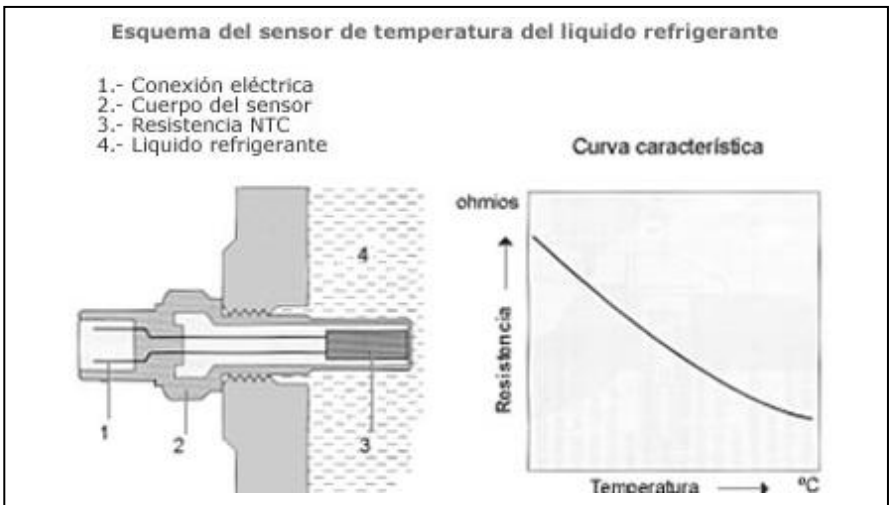


IMAGEN 11: Esquema sensor de temperatura líquido refrigerante.

La resistencia de medición forma parte de un circuito divisor de tensión alimentado con 5 V. La tensión que se mide en la resistencia depende, por tanto, de la temperatura. Ésta se lee a través de un convertidor analógico-digital y es una medida de la temperatura del sensor. La unidad de control del motor tiene almacenada una curva característica que indica la temperatura correspondiente a cada valor de resistencia o tensión de salida.

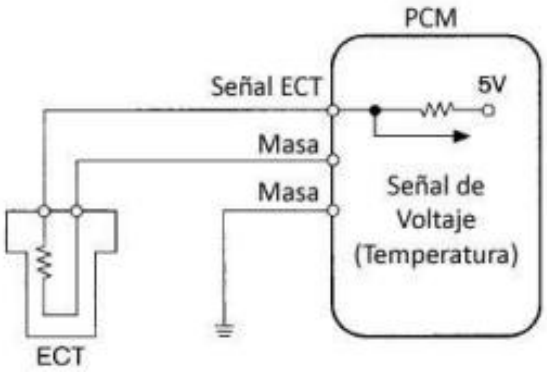


IMAGEN 12: Esquema eléctrico STM

El sensor de temperatura del motor se encuentra situada próximo a la conexión de la manguera del agua del radiador.

IMAGEN 13: Localización STM

El fallo de este sensor puede causar diferentes problemas como problemas de arranque ya sea con el motor en frío o en caliente y consumo en exceso del combustible.

Puede ocasionar además que el ventilador este continuamente encendido o bien problemas de sobrecalentamiento del motor.

5.1.2.3 HARDWARE DE COMUNICACIÓN

El soporte hardware de comunicación del cual se dispondrá, procederá del departamento de ingeniería electrónica. Se trata de dos PCB's ya diseñadas que incorporan un transmisor/receptor de radiofrecuencia con conector para el convertidor RS 232 Serial.

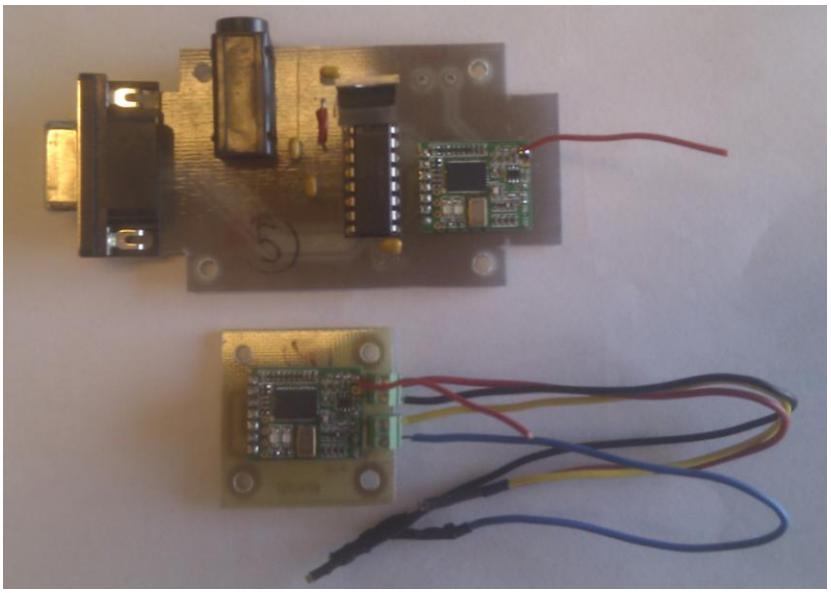


IMAGEN 14: Hardware de comunicación

6. JUSTIFICACIÓN DE LA SOLUCIÓN ADOPTADA

La realización del diseño y cálculo del equipo se ha estructurado en los subsistemas existentes, eléctrico y electrónico.

6.1 SUBSISTEMA ELÉCTRICO

6.1.1 FUENTE DE ALIMENTACIÓN (BATERÍA)

La batería del vehículo será la encargada de proporcionar la principal fuente de alimentación de todo el sistema. Esta será la batería Bosch S6, batería con potencia para los vehículos más exigentes con un elevado consumo energético.

6.1.2 CABLEADO DE CONEXIÓN DE PINES

El cableado empleado para la interconexión de todos los pines con los diferentes sensores y elementos de comunicación, se utilizará cables Dupont flexibles de 2.54mm redondos, con uno de los extremos con jumper tipo hembra de crimp.

6.1.3 CONVERTIDOR SERIE RS232

El Cable Adaptador USB a Puerto Serie RS232 DB9 permite conectar dispositivos seriales RS232 DB9 al PC a través de un puerto USB. Una solución eficaz y económica que permite acortar la distancia en términos de compatibilidad entre ordenadores modernos y periféricos preexistentes con conectividad serial.

6.1.4 CABLE TRANSMISIÓN DE DATOS

A través de este cable se realizará la transmisión de datos de programación del PC al micro controlador. Este cable está formado por un terminal USB estándar en uno de los extremos que conectará con el PC y en el otro un terminal mini USB que se conectará a la placa. Con este mismo cable, podremos alimentar la placa con una tensión de 3.3V, tensión procedente del PC.

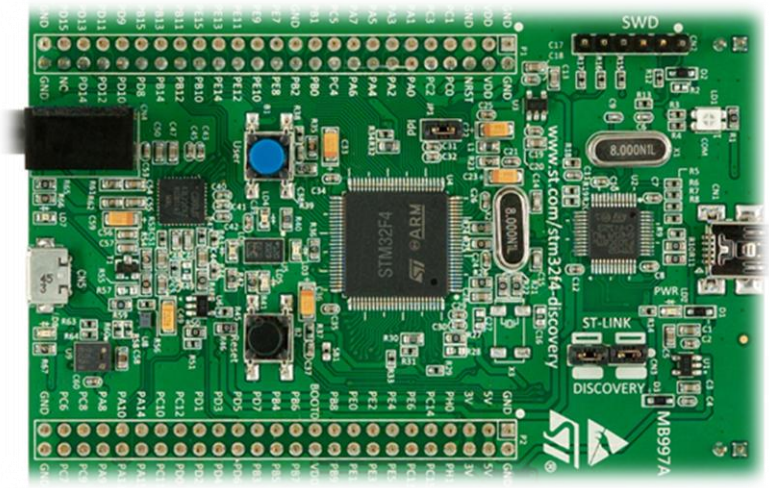
6.2 SUBSISTEMA ELECTRÓNICO

6.2.1 PROCESADOR

A través del kit de placa electrónica constituida por un micro controlador STM32F407VGT6 con Núcleo ARM Cortex-M4F 32-bit, 1 MB Flash, 192 KB de RAM, se procesará todo el código del sistema. Este kit dispone de todo lo necesario como pueden ser:

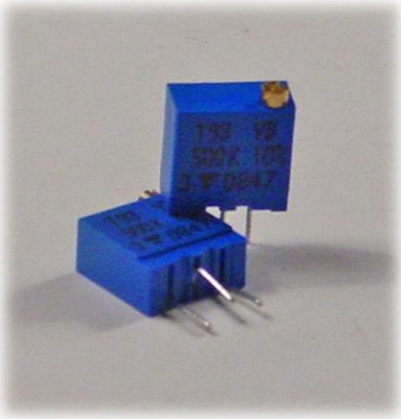
- Pines entrada/salida configurables para analógico/digital.
- Pines dedicados a la conversión de las señales analógicas procedentes de los sensores (potenciómetros) a señales digitales.
- Pines dedicados para la comunicación serie USART.
- Pines de salida de tensión a 3V y 5V
- Pines de masa y Vcc para alimentación externa.

Y otras muchas funciones que no se emplean en este sistema de telemetría.



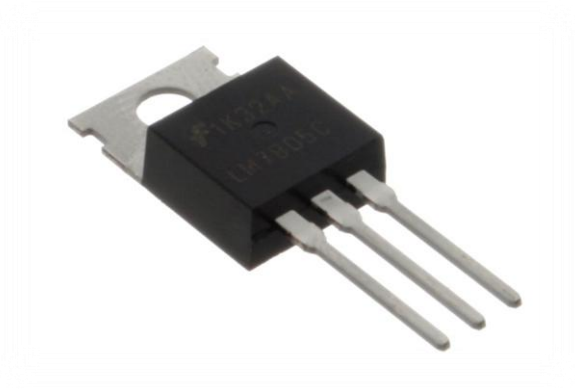
6.2.2 COMPONENTES DE SENSADO

Se ha decidido emplear potenciómetros multivueltas, puesto que nos permiten la generación variable de señales analógicas y simular de la manera más económica y práctica lo que serían sensores de temperatura, presión, etc. Estos potenciómetros son de 100KΩ.



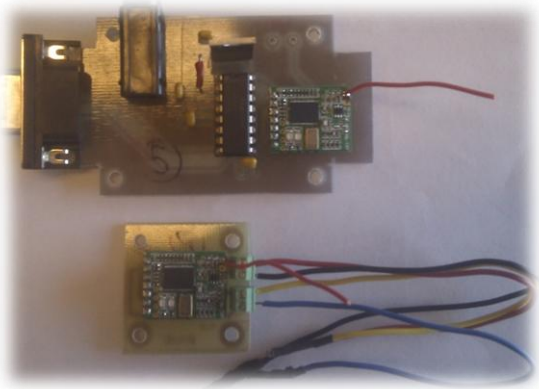
6.2.3 REGULADOR DE TENSIÓN LM7805

A través de este componente electrónico, regulador de tensión LM7805, se consigue una tensión estabilizada de 5V para la alimentación necesaria de la placa electrónica Discovery STM32F407VGT6.



6.2.4 HARDWARE DE RADIOFRECUENCIA

Este hardware será el encargado de realizar de forma correcta la transmisión de datos serie a través de radiofrecuencia. Actúan ambos con transmisor/receptor.



7. SOFTWARE

Aquí se muestra detalladamente línea a línea todo el código empleado para la realización del sistema.

7.1 PROGRAMACIÓN DEL MICRO CONTROLADOR

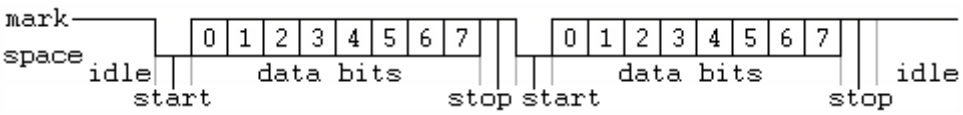
Para la configuración del ADC, puesto que el de la placa es de 12 bits y nuestra señal sensada puede oscilar entre 0V y 3V, si realizamos los cálculos pertinentes vemos que:

$$valor\ máximo = 2^{12} = 4096$$

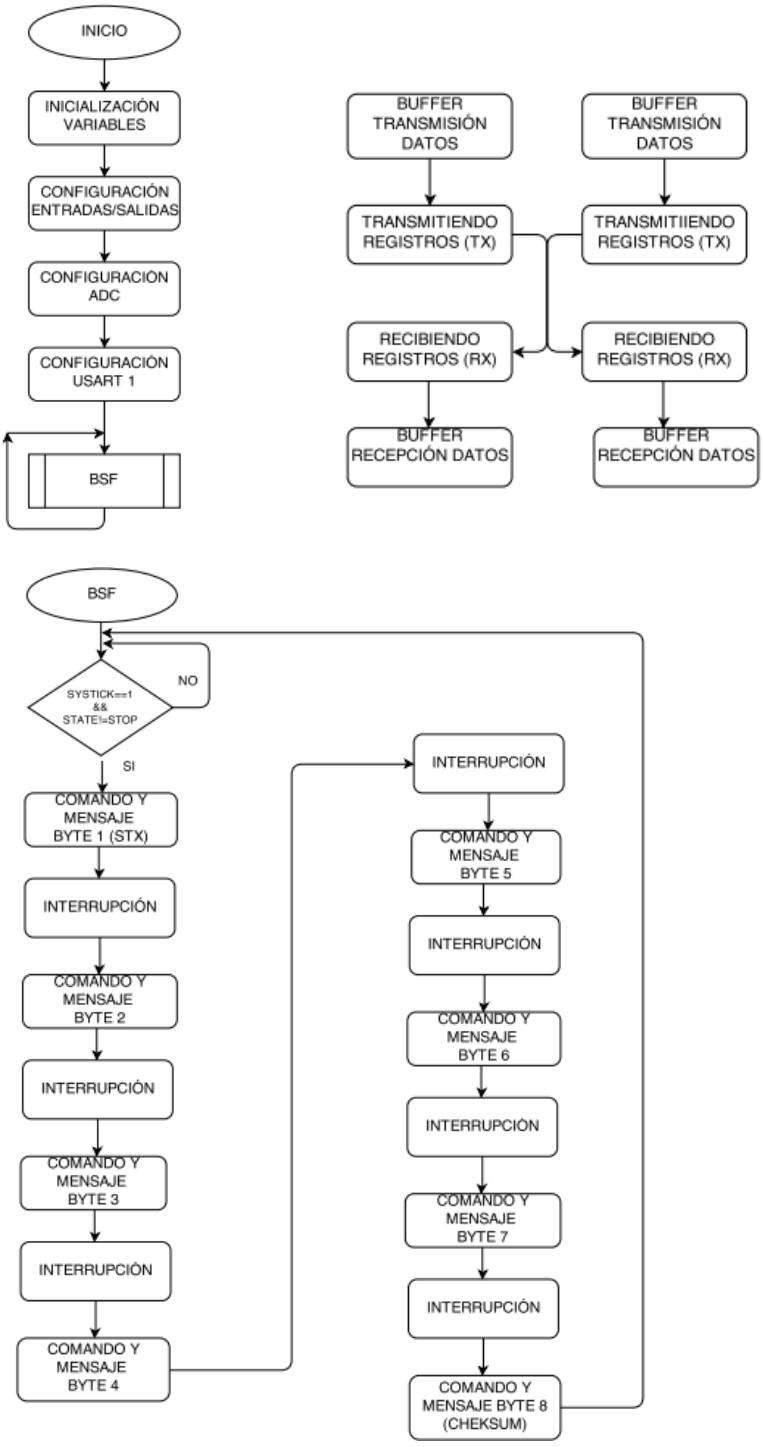
Este valor nos indica que necesitaríamos más de 12 bits, por tanto necesitaremos trocear cada uno de los datos en parte alta y en parte baja, es decir, cada dato/byte constara de 8 bits.

Para establecer la comunicación serie, se envía en primer lugar una señal inicial anterior al primer bit de cada byte, carácter o palabra codificada. Una vez enviado el código correspondiente, se envía inmediatamente una señal de stop después de cada palabra codificada.

La señal de inicio (start) sirve para preparar al mecanismo de recepción o receptor, la llegada y registro de un símbolo, mientras que la señal de stop sirve para predisponer al mecanismo de recepción para que tome un descanso y se prepare para la recepción del nuevo símbolo.



7.1.1 FLUJOGRAMAS



7.1.2 MAIN.C

```
#include "stm32f4xx.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_usart.h"
#include "misc.h"
#include "BoardConfig.h"
#include "stdio.h"
#include "stm32f4xx_adc.h"
#include "stdarg.h"
#include "time.h"

// Constantes del programa
#define SEND_BUFFER_SIZE 1
#define START_TOKEN 'A'      /*orden de empezar la transmisión*/
#define STOP_TOKEN 'C'       /*Orden de parar la transmisión*/
#define MVOLT_REF    3000    /* tension de referencia en mV, tensión de fondo de escala */


// Estados de la maquina del programa
#define STATE_WAITING 0   /*estado de espera*/
#define STATE_SENDING 1   /*estado de funcionamiento/envío*/
#define STATE_STOP 2      /*estado de paro*/




// variables Globales
signed int bufferTransmit[SEND_BUFFER_SIZE];
int counter, state;
int luz, temp;
int press;


/*programa principal*/

int main(void)
{
 int i;
unsigned int retraso;
 // Setup all peripherals
C BoardInit();
 luxes_inicializar();

      //limpieza del buffer
      for (i=0;i<SEND_BUFFER_SIZE;i++)bufferTransmit[i]=0;

      state = STATE_STOP;

      while(1)
    {
```

```
                // if (System tick has occurred) and (state != STOP)
                if (((((SysTick->CTRL)>>16)&0x1) == 1) && (state != STATE_STOP))
                      {

                      luz=voltios_leer_luz();
                      temp=voltios_leer_temp();
                      press=voltios_leer_press();
                      // update sensor readings
                      // send first data for interrupt activation
                      bufferTransmit[0]=0xBB;//valor fijo 0xBB byte 1
                      USART_SendData(USART1,bufferTransmit[0]);//envio

                      for(retraso=0;retraso<5000;retraso++); */bucle retraso*/

                      bufferTransmit[0]=luz/256;//parte alta byte 2
                      USART_SendData(USART1,bufferTransmit[0]); //envío

                      for(retraso=0;retraso<5000;retraso++);  */bucle retraso*/

                      bufferTransmit[0]=luz%256; //parte baja byte 3
                      USART_SendData(USART1,bufferTransmit[0]); //envío

                      for(retraso=0;retraso<5000;retraso++);  */bucle retraso*/

                      bufferTransmit[0]=temp/256; parte alta byte 4
                      USART_SendData(USART1,bufferTransmit[0]); //envío

                      for(retraso=0;retraso<5000;retraso++);*/bucle retraso*/

                      bufferTransmit[0]=temp%256; parte baja byte 5
                      USART_SendData(USART1,bufferTransmit[0]); //envío

                      for(retraso=0;retraso<5000;retraso++);*/bucle retraso*/

                      bufferTransmit[0]=press/256; parte alta byte 6
                      USART_SendData(USART1,bufferTransmit[0]); //envío

                      for(retraso=0;retraso<5000;retraso++);*/bucle retraso*/

                      bufferTransmit[0]=press%256; parte baja byte 7
                      USART_SendData(USART1,bufferTransmit[0]); //envío

                      for(retraso=0;retraso<5000;retraso++);*/bucle retraso*/


                      bufferTransmit[0]=((luz/256)+(luz%256)+(temp/256)+(temp%25
                      6)+(press/256)+(press%256))%256; //parte checksum byte 8
                      USART_SendData(USART1,bufferTransmit[0]); //envío

                      }

          }

}
```

```
void USART1_IRQHandler(void)
{
	uint8_t temprx;

	if(USART_GetFlagStatus(USART1,USART_FLAG_TC) == 1)// if an Tx transmit
is complete
	{
		if (state != STATE_STOP)
		{
			if(counter == SEND_BUFFER_SIZE-1) // whole packet send?
			{
				counter = 0;
				state = STATE_WAITING;
			}
			else // keep sending rest of packet
			{
				USART_SendData(USART1,bufferTransmit[counter+1]);
				counter++;
			}
		}
		USART_ClearFlag(USART1,USART_FLAG_TC);
	}

	if(USART_GetFlagStatus(USART1,USART_FLAG_RXNE) == 1)// data in RX
	{
		temprx = USART_ReceiveData(USART1);
		// start token?
		if((temprx== START_TOKEN) && (state == STATE_STOP))
		{
			state = STATE_SENDING;
			counter = 0;
		}

		// stop request?
		if((temprx == STOP_TOKEN) && (state != STATE_SENDING))
			state = STATE_STOP;

		// clear flag
		USART_ClearFlag(USART1,USART_FLAG_RXNE);
	}
}

//Leer caida de tensión provocada por la LDR en estado de oscuridad
//@returns voltios según luz

uint32_t voltios_leer_luz(void)
{

volatile static uint32_t valor_adc=0;
volatile static uint32_t milivoltios=0;

ADC_ClearFlag(ADC1,ADC_FLAG_JEOC);	// borrar flag de fin conversion
```

```
ADC_SoftwareStartInjectedConv(ADC1);    // inicial conversion

while (ADC_GetFlagStatus(ADC1,ADC_FLAG_JEOC) == RESET); // esperar conversion

valor_adc = ADC_GetInjectedConversionValue(ADC1, ADC_InjectedChannel_1);

milivoltios = (valor_adc*MVOLT_REF)/0xFFF; // 12 bits ADC

      return milivoltios;
}

uint32_t voltios_leer_temp(void)
{
      volatile static uint32_t valor_adc=0;
      volatile static uint32_t milivoltios=0;

      ADC_ClearFlag(ADC2,ADC_FLAG_JEOC);      // borrar flag de fin
conversion

      ADC_SoftwareStartInjectedConv(ADC2);    // inicial conversion

      while (ADC_GetFlagStatus(ADC2,ADC_FLAG_JEOC) == RESET); // esperar
conversion

      valor_adc = ADC_GetInjectedConversionValue(ADC2,
ADC_InjectedChannel_1);

      milivoltios = (valor_adc*MVOLT_REF)/0xFFF; // 12 bits ADC

            return milivoltios;
}

uint32_t voltios_leer_press(void)
{
      volatile static uint32_t valor_adc=0;
      volatile static uint32_t milivoltios=0;

      ADC_ClearFlag(ADC3,ADC_FLAG_JEOC);      // borrar flag de fin
conversion

      ADC_SoftwareStartInjectedConv(ADC3);    // inicial conversion

      while (ADC_GetFlagStatus(ADC3,ADC_FLAG_JEOC) == RESET); // esperar
conversion

      valor_adc = ADC_GetInjectedConversionValue(ADC3,
ADC_InjectedChannel_1);

      milivoltios = (valor_adc*MVOLT_REF)/0xFFF; // 12 bits ADC

            return milivoltios;
}
```

7.1.3 BOARDCONFIG.C

```
#include "stm32f4xx.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_usart.h"
#include "misc.h"
#include "BoardConfig.h"
#include "stdio.h"
#include "stm32f4xx_adc.h"
#include "voltios_ldr.h"
#include "stdlib.h"
#include "stdarg.h"

GPIO_InitTypeDef GPIO_InitStructure;
USART_InitTypeDef USART_InitStructure;

void BoardInit (void)
{
	// setup uart
	configureUART();
	// configure leds
	configureLEDS();

		// Setup SystemTick timer
	SysTick->LOAD = (72000 / 8) * TIMER_PERIODE_MS;
	SysTick->VAL = 0;
	SysTick->CTRL = 0x1;
}

void configureUART (void)
{
	/* enable usart clock */
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
	/* enable gpio uart clock */
	RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
	RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);

	/* Configure the USART */
	GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_10;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
	GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
	GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
	GPIO_Init(GPIOA, &GPIO_InitStructure);
	GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_6;
	GPIO_Init(GPIOB, &GPIO_InitStructure);
	GPIO_PinAFConfig(GPIOB, GPIO_PinSource6, GPIO_AF_USART1);
//PB6(AF)TX
	GPIO_PinAFConfig(GPIOA, GPIO_PinSource10, GPIO_AF_USART1);
//PA10(AF)RX

	/* Configure USART1 settings */
	USART_InitStructure.USART_BaudRate = 9600;
```

```
	USART_InitStructure.USART_WordLength = USART_WordLength_8b;
	USART_InitStructure.USART_StopBits = USART_StopBits_1;
	USART_InitStructure.USART_Parity = USART_Parity_No;
	USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
	USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
	USART_Init(USART1, &USART_InitStructure);
	USART_Cmd(USART1, ENABLE);
	NVIC_EnableIRQ(USART1_IRQn);
	USART1->CR1 |= 0x1<<6;// inable interupt op Transmission complete
	USART1->CR1 |= 0x1<<5;// inable interupt op Data receive not empty
}

void configureLEDS (void)
{
	/* Enable the GPIO_LED Clock */
	RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOE, ENABLE);

	/* Configure the LEDS */
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_15;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
	GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
	GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOE, &GPIO_InitStructure);

	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_14;
	GPIO_Init(GPIOE, &GPIO_InitStructure);
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;
	GPIO_Init(GPIOE, &GPIO_InitStructure);
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
	GPIO_Init(GPIOE, &GPIO_InitStructure);
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
	GPIO_Init(GPIOE, &GPIO_InitStructure);
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
	GPIO_Init(GPIOE, &GPIO_InitStructure);
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
	GPIO_Init(GPIOE, &GPIO_InitStructure);
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
	GPIO_Init(GPIOE, &GPIO_InitStructure);
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
	GPIO_Init(GPIOE, &GPIO_InitStructure);
}

void luxes_inicializar(void) {

    ADC_InitTypeDef       ADC_InitStructure;
    ADC_CommonInitTypeDef ADC_CommonInitStructure;

    /* Puerto C ------------------------------------------------------------
*/

    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
```

```
    /* PC1|PC3|PC0 para entrada analógica,libre */

    GPIO_StructInit(&GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Pin= GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AN;
    GPIO_InitStructure.GPIO_PuPd=GPIO_PuPd_NOPULL ;
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin= GPIO_Pin_3;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Pin= GPIO_Pin_0;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    /* Activar ADC1, ADC2, ADC3------------------------------------------
------------*/

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC2, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC3, ENABLE);

    /* ADC Common Init ---------------------------------------------------
-*/

    ADC_CommonStructInit(&ADC_CommonInitStructure);
    ADC_CommonInitStructure.ADC_Mode                = ADC_Mode_Independent;
    ADC_CommonInitStructure.ADC_Prescaler           = ADC_Prescaler_Div4; //
max 30 MHz segun datasheet
    ADC_CommonInitStructure.ADC_DMAAccessMode       =
ADC_DMAAccessMode_Disabled;
    ADC_CommonInitStructure.ADC_TwoSamplingDelay    =
ADC_TwoSamplingDelay_5Cycles;
    ADC_CommonInit(&ADC_CommonInitStructure);

    /* ADC Init ----------------------------------------------------------
--*/

    ADC_StructInit (&ADC_InitStructure);
    ADC_InitStructure.ADC_Resolution             = ADC_Resolution_12b;
    ADC_InitStructure.ADC_ScanConvMode           = DISABLE;
    ADC_InitStructure.ADC_ContinuousConvMode     = DISABLE;
    ADC_InitStructure.ADC_ExternalTrigConvEdge   =
ADC_ExternalTrigConvEdge_None;
    ADC_InitStructure.ADC_DataAlign              = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfConversion        = 1;
    ADC_Init(ADC1, &ADC_InitStructure);
    ADC_Init(ADC2, &ADC_InitStructure);
    ADC_Init(ADC3, &ADC_InitStructure);

    /* Establecer la configuración de conversión PC1 ---------------------
------*/

    ADC_InjectedSequencerLengthConfig(ADC1, 2);
    ADC_SetInjectedOffset(ADC1, ADC_InjectedChannel_1, 90);
    ADC_InjectedChannelConfig(ADC1, ADC_Channel_11, 1,
ADC_SampleTime_480Cycles);
```

```
    /* Establecer la configuración de conversión PC3-------------------------
-----*/

       ADC_InjectedSequencerLengthConfig(ADC2, 2);
       ADC_SetInjectedOffset(ADC2, ADC_InjectedChannel_1, 90);
       ADC_InjectedChannelConfig(ADC2, ADC_Channel_13, 1,
ADC_SampleTime_480Cycles);

    /* Establecer la configuración de conversión PC0-------------------------
-----*/

          ADC_InjectedSequencerLengthConfig(ADC3, 2);
          ADC_SetInjectedOffset(ADC3, ADC_InjectedChannel_1, 90);
          ADC_InjectedChannelConfig(ADC3, ADC_Channel_10, 1,
ADC_SampleTime_480Cycles);
       /* Poner en marcha ADC ---------------------------------------------
-----*/

    ADC_Cmd(ADC1, ENABLE);
    ADC_Cmd(ADC2, ENABLE);
    ADC_Cmd(ADC3, ENABLE);
}
```

7.1.4 BOARDCONFIG.H

```
void BoardInit (void);
void configureUART (void);
void configureLEDS (void);
void luxes_inicializar(void);
uint32_t voltios_leer_luz(void);
uint32_t voltios_leer_temp(void);
uint32_t voltios_leer_press(void);

#define TIMER_PERIODE_MS 300
```

7.2 PROGRAMACIÓN INTERFAZ DE USUARIO

```
function varargout = MATLAB_COM_SERIE(varargin)

% inicialización del codigo
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MATLAB_COM_SERIE_OpeningFcn, ...
                   'gui_OutputFcn',  @MATLAB_COM_SERIE_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% fin inicialización codigo - DO NOT EDIT


% --- Executes just before MATLAB_COM_SERIE is made visible.
function MATLAB_COM_SERIE_OpeningFcn(hObject, eventdata, handles,
varargin)


global usart1 x y contador contador1 x1 y1 contador2 x2 y2

clc;
contador=0;x=0;y=0;contador1=0;x1=0;y1=0;contador2=0;x2=0;y2=0;

%Valores por defecto
handles.puerto='COM3';
set(handles.popCom,'Value',3);
handles.velocidad=9600;
set(handles.popVelocidad,'Value',3);

% Configuración del puerto serie
usart1 = serial(handles.puerto);
set(usart1,'BaudRate',handles.velocidad);
set(usart1,'DataBits',8);
set(usart1,'Parity','none');
set(usart1,'StopBits',1);
set(usart1,'FlowControl','none');
set(usart1,'BytesAvailableFcnCount', 2); % Se configura en n° de bytes
que debe haber en el buffer de recepción para disparar el evento
Rx_Callback
set(usart1, 'BytesAvailableFcnMode' ,'byte');
set(usart1,'BytesAvailableFcn',{@Rx_Callback,handles});
fopen(usart1);

% Choose default command line output for MATLAB_COM_SERIE
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);




% --- Outputs from this function are returned to the command line.
function varargout = MATLAB_COM_SERIE_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
```

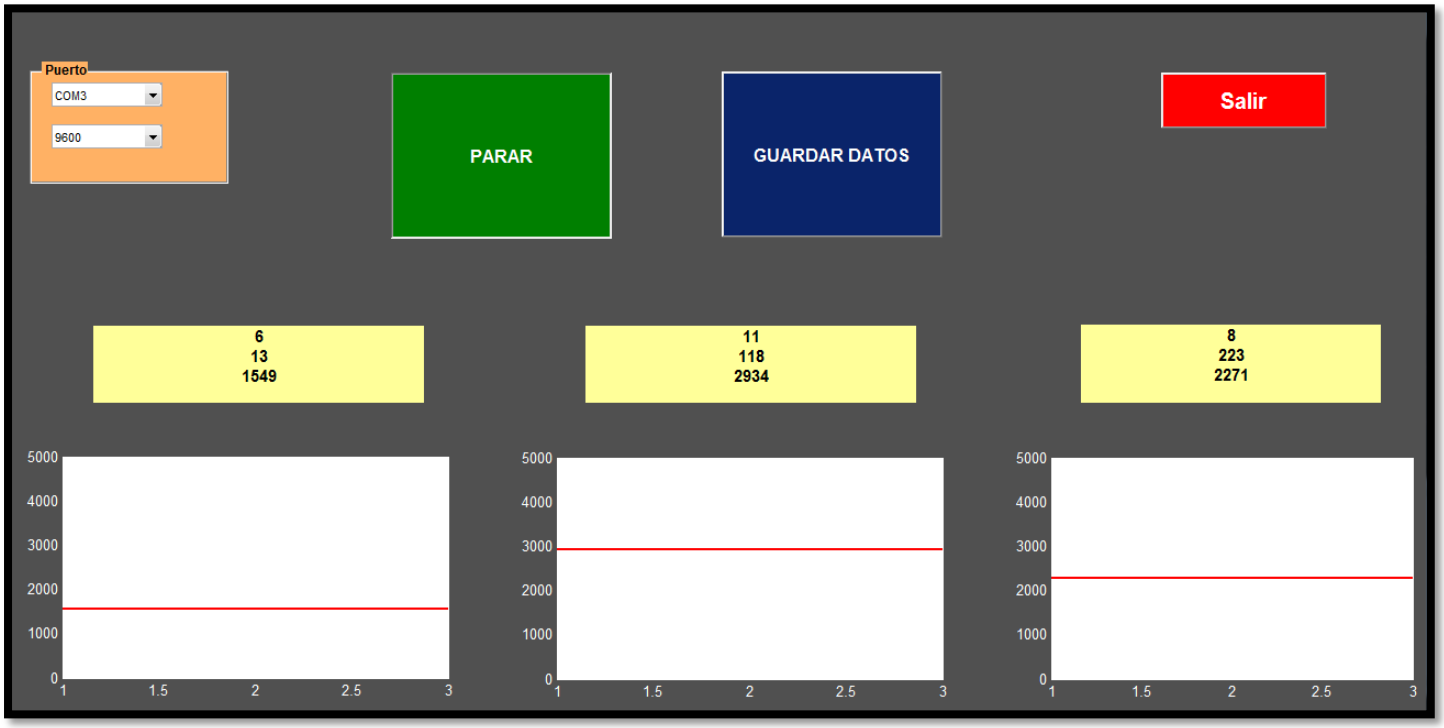
```
% --- Executes on selection change in popCom.
function popCom_Callback(hObject, eventdata, handles)
% hObject    handle to popCom (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popCom
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popCom
contents = get(hObject,'String');
selectedText = contents{get(hObject,'Value')};
handles.puerto = selectedText;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function popCom_CreateFcn(hObject, eventdata, handles)

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% ejecuta la function para seleccionar la velocidad
function popVelocidad_Callback(hObject, eventdata, handles)
% Hints: contents = cellstr(get(hObject,'String')) returns
popVelocidad contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popVelocidad
contents = get(hObject,'String');
selectedText = contents{get(hObject,'Value')};
handles.velocidad = str2double(selectedText);
guidata(hObject,handles)


% muestra visualmente el cambio de velocidad en el selector
desplegable
function popVelocidad_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Rx_Callback(hObject, eventdata, handles)
global usart1 contador x y contador1 x1 y1 contador2 x2 y2

num=usart1.BytesAvailable;
if num>0
    dato(1)=fread(usart1,1);% byte 1, dato fijo BB
    dato1(1)=fread(usart1,1);% byte 2
```

```
    dato1(2)=fread(usart1,1);% byte 3
    dato1(3)=dato1(1)*256+dato1(2);
    dato2(1)=fread(usart1,1);% byte 4
    dato2(2)=fread(usart1,1);% byte 5
    dato2(3)=dato2(1)*256+dato2(2);
    dato3(1)=fread(usart1,1);% byte 6
    dato3(2)=fread(usart1,1);% byte 7
    dato3(3)=dato3(1)*256+dato3(2)
    dato(1)=fread(usart1,1);% byte 8
    contador=contador+1;
    x(contador)=contador;
    y(contador)=dato1(3);
    contador1=contador1+1;
    x1(contador1)=contador1;
    y1(contador1)=dato2(3);
    contador2=contador2+1;
    x2(contador2)=contador2;
    y2(contador2)=dato3(3);
    plot(handles.pltGrafica,x,y,'-r','LineWidth',2);
    plot(handles.pltGrafica1,x1,y1,'-r','LineWidth',2);
    plot(handles.pltGrafica2,x2,y2,'-r','LineWidth',2);
    set(handles.lblMensaje,'String',dato1);
    set(handles.lblMensaje1,'String',dato2);
    set(handles.lblMensaje2,'String',dato3);
    clc;
end


% ejecuta el boton salir y te pregunta
function cmdSalir_Callback(hObject, eventdata, handles)

global usart1

opc=questdlg('¿Desea salir del programa?','SALIR','Si','No','No');
if strcmp(opc,'No')
    return;
end

fclose(usart1);
delete(usart1);
clear usart1;

%clear
%clc
close all


% Ejecuta o para el programa al pulsar 'A' o 'C'
function tglMuestreo_Callback(hObject, eventdata, handles)
global usart1

isDown = get(hObject,'Value');

if isDown
```

```
    set(handles.tglMuestreo, 'string', 'PARAR')

    fwrite(usart1,'A');pause(0.2);
else
    set(handles.tglMuestreo, 'string', 'MUESTREAR')

    fwrite(usart1,'C');pause(0.2);
end


Guarda los datos recibidos en un fichero de texto.txt y en una hoja
excel.
function cmdDatos_Callback(hObject, eventdata, handles)
global x y x1 y1 x2 y2

sal=[x;y;x1;y1;x2;y2];
fid = fopen('salida.txt','w');
fprintf(fid,'%4.0f  %2.1f\r\n',sal);
fclose(fid);
xlswrite('salida.xlsx',sal');

function tglMuestreo_ButtonDownFcn(hObject, eventdata, handles)

function pltGrafica1_CreateFcn(hObject, eventdata, handles)
```

7.2.1 INTERFAZ DE USUARIO

Está imagen muestra el interfaz de usuario, a través del cual controlaremos y visualizaremos las señales obtenidas de las medidas sensadas.



Documento 2

Pliego de Condiciones

Índice

1. Objeto 3
2. Descripción general del montaje 3
3. Condiciones y normas de carácter general 4
4. Condiciones particulares 4
4.1 Condiciones técnicas de los materiales 4
4.1.1 Placa electrónica 5
4.1.2 Circuito RF 5
4.1.3 Cableado 5
4.1.3.1 Cableado de conexión de los pines 5
4.1.3.2 Cable de transmisión de datos 6
4.1.3.3 Cable de convertidor Serial RS232 6
4.1.4 Potenciómetros 7
4.1.5 Regulador de tensión 7
4.1.6 Ordenador Pc 7
5. Condiciones técnicas de la ejecución 8
5.1 Introducción 8
5.2 Montaje del circuito de pruebas 8
5.3 Control de calidad 13
5.4 Pruebas y ajustes finales 13

PLIEGO DE CONDICIONES

1. Objeto

La presente especificación técnica se refiere al conjunto de trabajos y exigencias a seguir para la ejecución del SISTEMA DE MONITORIZACIÓN INALÁMBRICA PARA MOTORES DE COMPETICIÓN VÍA USART.

Entre las exigencias se encuentran las condiciones técnicas que deben cumplir los materiales seleccionados que forman parte del sistema del proyecto. Otra de las partes que engloba este documento son las normas y reglamentaciones legales que afectan a los componentes que existen en el mercado, sus valores nominales, y ensayos pasados para regular el control de calidad. Para una mayor información sobre los componentes utilizados se consultarán las hojas de características de los mismos que se adjuntan en el Anexo "Datasheets de los componentes".

El objeto final de este proyecto es el estudio, diseño e implementación de la parte electrónica e informática de un sistema de adquisición de datos inalámbrico a tiempo real (telemetría), en el cual se tomarán mediciones de la temperatura del motor, posición del pedal del acelerador y caudal de aire de entrada al motor. Además, la medida debe ser fiable en un rango determinado.

2. Descripción general del montaje

Para la realización física del proyecto se deberán seguir unos determinados pasos que se cumplirán bajo este orden:

- Adquisición del material necesario.
- Instalación en el PC de las aplicaciones.
- Grabación del programa del microcontrolador.
- Pruebas en taller.
- Montaje de los componentes y del cableado.
- Comprobación del correcto montaje de la placa y componentes.
- Puesta en funcionamiento del conjunto y ajustes oportunos.

Todas las partes que en conjunto forman la obra del presente proyecto deberán ser ejecutadas por personal técnico competente, sometiéndose a las normas y reglas

previstas para este tipo de montajes, no haciéndose cargo el proyectista de su incumplimiento.

3. Condiciones y normas de carácter general

En el presente documento se ha especificado la referencia de los componentes; si bien el encargado de montar la placa, por cualquier dificultad, no pudiera hacer uso de alguno de los componentes en este pliego de condiciones, procurará sustituirlo por otro de similares características, siempre con el conocimiento y el consentimiento del autor del proyecto, pasando la responsabilidad de un posible mal funcionamiento a la persona que haya realizado dicha modificación no autorizada.

La verificación de los componentes que se van a emplear en el proyecto no será necesaria ya que se darán por buenas las especificaciones que proporcionan los fabricantes en sus hojas de características.

Además de las características técnicas de cada componente, que se redactarán en el apartado pertinente, durante la ejecución del proyecto deberán cumplirse las siguientes normas:

- Ordenanza general de seguridad e higiene en el trabajo.
- Y en general, se deberán cumplir todas las normas que tengan alguna relación directa con el proyecto.

4. Condiciones particulares

4.1 Condiciones técnicas de los materiales

El presente texto se refiere exclusivamente al conjunto de trabajos necesarios para el montaje del diseño eléctrico y electrónico y puesta a punto del mismo, así como de la realización de las pruebas pertinentes para dejar todo el conjunto funcionando correctamente.

Las características y propiedades de todos los materiales empleados en la realización del proyecto como valores, tolerancias, curvas de características, etc…, vienen detalladas claramente en las hojas de características facilitadas por los respectivos fabricantes que se adjuntan en el Anexo “Datasheets de los componentes”, exceptuando algunos componentes de los más comunes como pueden ser los potenciómetros y condensadores.

Por tanto, se comentarán las características de montaje, así como otras características que no estén suficientemente desarrolladas en las hojas de características de cada componente, analizando cada componente por separado. Por todo esto se omite el apartado de control de calidad ya que se supone que todos los componentes han pasado los test indicados en las hojas de características.

4.1.1 Placa electrónica

La placa electrónica tendrá las características y dimensiones especificadas por el fabricante (ver datasheet del fabricante). La placa utilizada será una placa de tipo kit comandada por un microcontrolador de la familia ARM Cortex M4.

La placa será encargada a alguna empresa de distribución de microcontroladores de reconocido prestigio.

4.1.2 Circuito RF

Estos circuitos son 2, distribuidos en 2 placas de dimensiones 32x32mm y 90x45mm. Las placas han sido proporcionadas por el departamento de ingeniería electrónica para sus pruebas y desarrollo.

Las placas utilizadas son unas placas de cobre virgen, con un espesor estándar.

Finalmente estás serán encargadas a una empresa.

4.1.3 Cableado

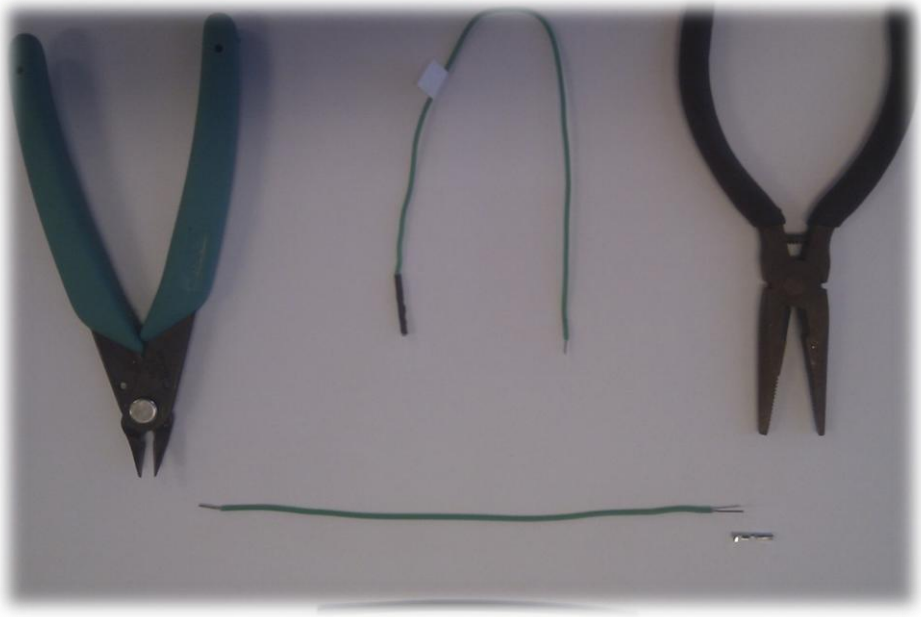
4.1.3.1 Cableado de conexión de los pines

Para el cableado de conexión entre los pines de la tarjeta y el resto de componentes, se ha optado para las pruebas la realización manual de estos debido a su fácil y rápido montaje. No obstante, para su final ejecución podrán ser comprados en cualquier tienda de componentes electrónicos como cables Dupont flexibles de 2.54mm con conector para pines de tipo hembra.

Para la realización manual se ha empleado lo siguiente:

- El cableado utilizado para la interconexión entre los pines de la tarjeta y PCB's, serán cables redondos de hilo de conexión <<Ariston>>, de sección 0.28mm^2.

- El conector que se utilizará en los extremos del cableado para la conexión de los pines será de tipo crimp de 2.54mm, y serán encargados al distribuidor inglés Farnell.
- Alicates de punta plana y de corte.
- Termoretractil.



4.1.3.2 Cable de transmisión de datos

El cable utilizado para la transmisión de datos de programación del Pc al microcontrolador, será un cable de conexión USB en uno de los extremos y conexión mini USB en el otro extremo. Este cable podrá ser solicitado al mismo distribuidor que el de la tarjeta de forma conjunta.

4.1.3.3 Cable de convertidor Serial RS232

Este cable será el utilizado para la conversión de datos serie. Su conexión se realizará a través de USB al ordenador y por medio de una conexión DB-9 al transmisor/receptor de RF.

Para su correcto funcionamiento, se facilita con el mismo un CD de drivers que tendrá que ser instalado en el Pc de uso.

Este convertidor RS232 podrá ser comprado a cualquier distribuidor de confianza de componentes electrónicos.

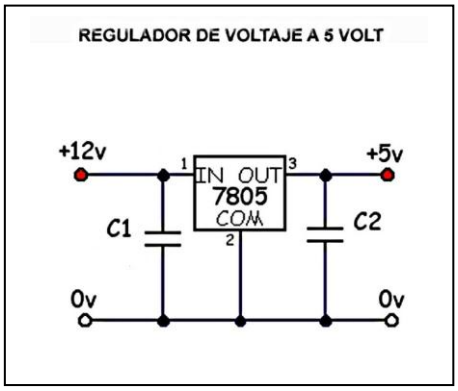
4.1.4 Potenciómetros

Los potenciómetros utilizados en la realización práctica, se trata de potenciómetros multivueltas de 100K con una tolerancia de 10%. Se han empleado este tipo de potenciómetros para poder cerciorarse ante cualquier cambio, de la sensibilidad del sistema.

4.1.5 Regulador de tensión

El regulador de tensión empleado es el L7805CV del fabricante ST (ver datasheet). Este será el encargado de regular de 12V a 5V la tensión procedente de la batería del vehículo para la alimentación de la Discovery.

En su montaje se requerirá del uso de dos condensadores electrolíticos de C1 = 330nF y C2 = 100nF.



4.1.6 Ordenador Pc

El ordenador que se empleará, podrá ser tanto un ordenador de sobremesa como un ordenador portátil. No requerirá de características técnicas muy exigibles, bastará con poder ser de un sistema operativo Windows Xp, puesto que los programas empleados para la creación del código y visualización así lo requieren.

El resto de componentes y materiales que integran el proyecto no merecen una especial descripción, ya que son fácilmente localizables en el comercio, si bien el montador deberá tener siempre en cuenta las premisas de fiabilidad, seguridad y durabilidad, optando siempre por marcas de reconocido prestigio.

5. Condiciones técnicas de la ejecución

5.1 Introducción

En el presente texto se indicarán los aspectos en los que se debe extremar la exactitud en los trabajos para conseguir el montaje adecuado y la correcta puesta en funcionamiento del sistema.

5.2 Montaje del circuito de pruebas

Como se comentó anteriormente en el apartado de materiales, las placas tanto de control como de RF serán encargadas a un distribuidor de componentes electrónicos especializado.

En este punto se describen en orden de ejecución los pasos a seguir para la realización correcta de la presente obra.

Adquisición de componentes

La compra de los materiales, componentes y equipos necesarios deberá ser realizada después de un proceso de comprobación de precios y tiempos de suministros de los distintos distribuidores.

La adquisición de componentes debe de realizarse con la suficiente antelación para que estén dispuestos para el momento en que se necesiten para su instalación.

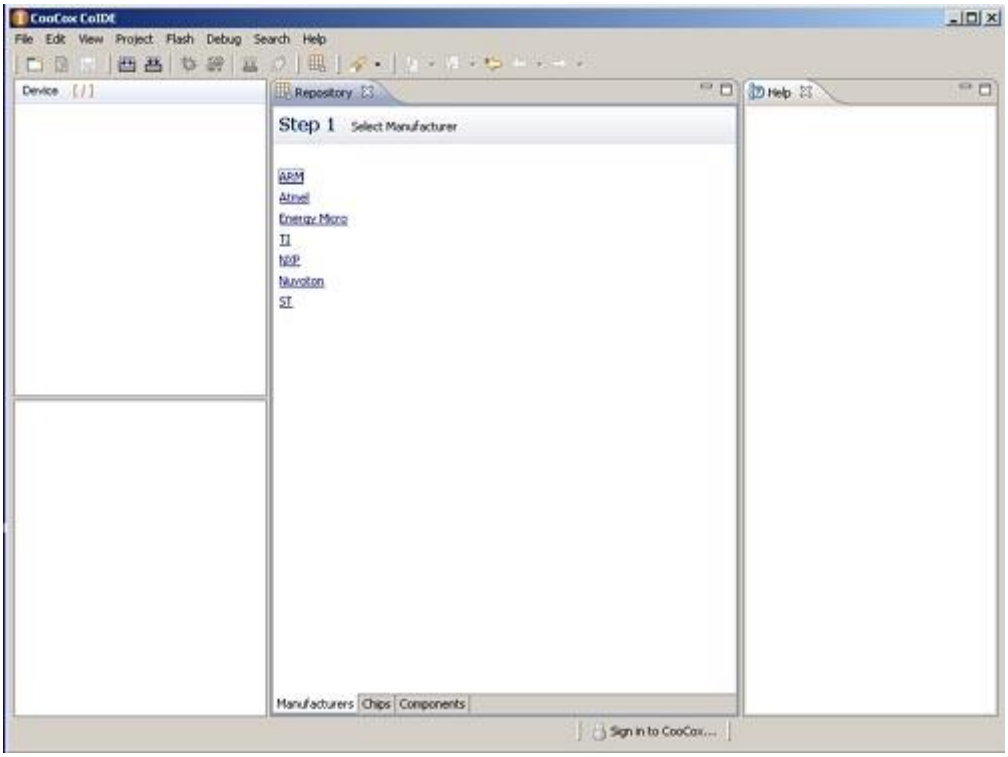
Si se realiza el montaje en cadena, se debe de tener en cuenta, el tener siempre disponible un stock mínimo, el cual puede cubrir alguna eventualidad no calculable.

Instalación de la aplicación de programación

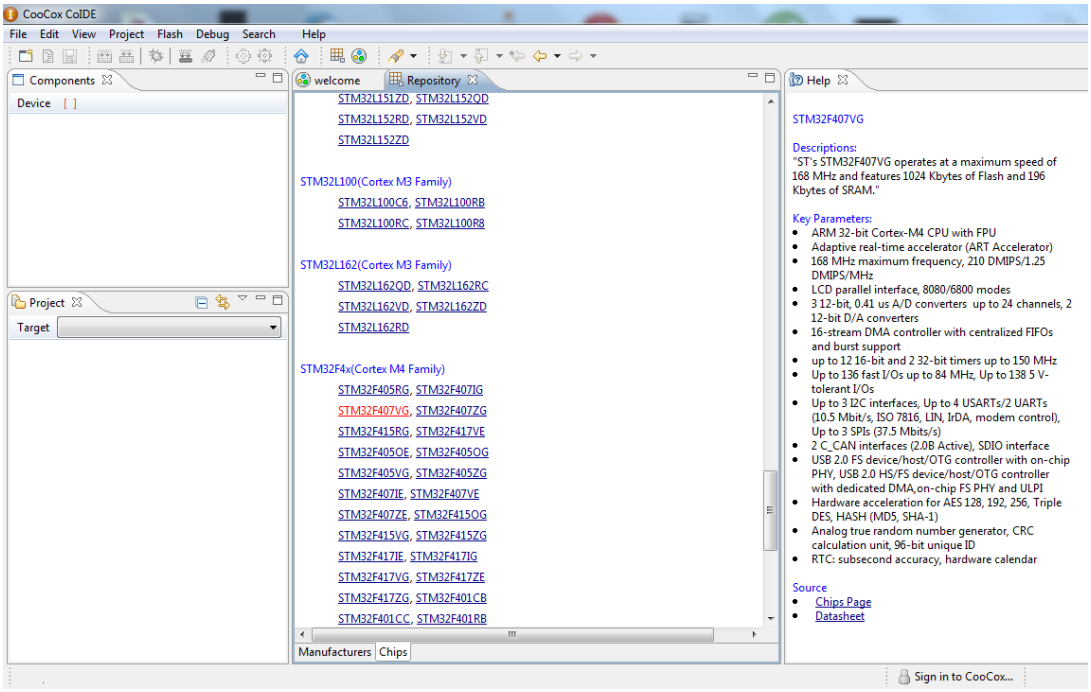
A continuación se mostrará de forma rápida e intuitiva la instalación del CooCox y adaptación para la tarjeta empleada.

1) vamos a la web de CooCox o lo descargamos directamente desde el enlace CoIDE (http://www.org/Tools/CoIDE.1.7.6.exe)
2) Lo instalamos en nuestro ordenador ejecutando el CoIDE-1.7.6.exe

Al arrancarlo por primera vez la primera pantalla que nos muestra es la siguiente:



Lo primero que hemos de hacer es seleccionar la marca concreta del microcontrolador que utilicemos, en nuestro caso ST.



Aquí nos aparecen los tipos de controladores de ST Microelectronics Cortex M4 disponibles. Seleccionamos el nuestro, el STM32F407VG. A la vez, observamos como en la ventana de la derecha nos muestra las características concretas de este microcontrolador:

STM32F407VG

Descriptions:

"ST's STM32F407VG operates at a maximum speed of 168 MHz and features 1024 Kbytes of Flash and 196 Kbytes of SRAM."

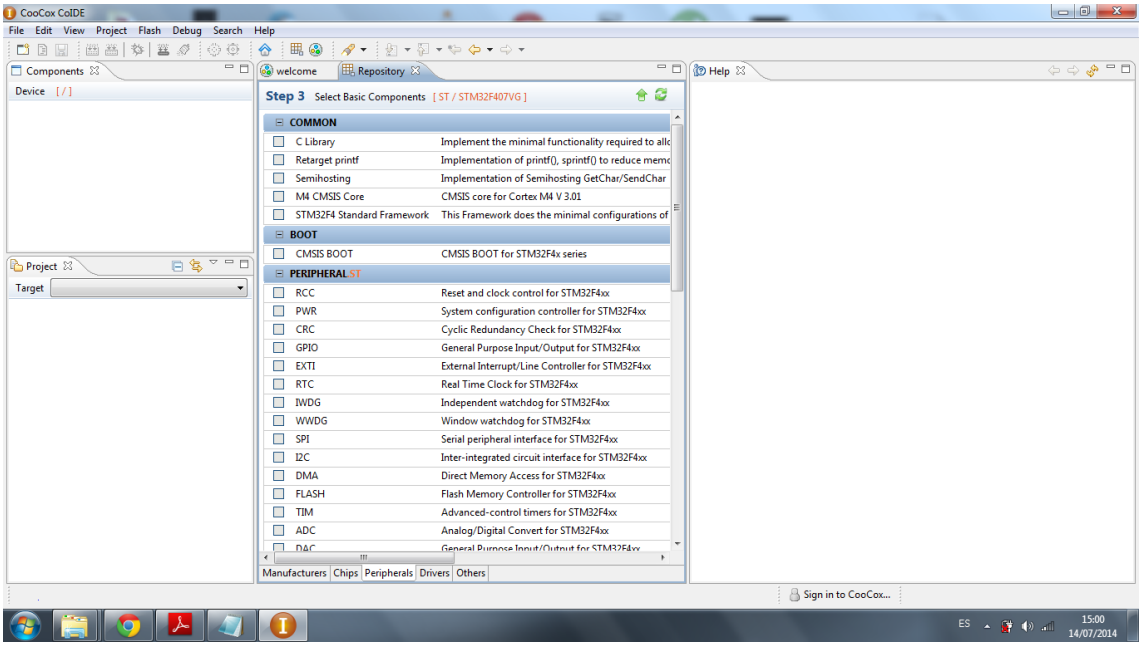
Key Parameters:

- ARM 32-bit Cortex-M4 CPU with FPU
- Adaptive real-time accelerator (ART Accelerator)
- 168 MHz maximum frequency, 210 DMIPS/1.25 DMIPS/MHz
- LCD parallel interface, 8080/6800 modes
- 3 12-bit, 0.41 us A/D converters up to 24 channels, 2 12-bit D/A converters
- 16-stream DMA controller with centralized FIFOs and burst support
- up to 12 16-bit and 2 32-bit timers up to 150 MHz
- Up to 136 fast I/Os up to 84 MHz, Up to 138 5 V-tolerant I/Os
- Up to 3 I2C interfaces, Up to 4 USARTs/2 UARTs (10.5 Mbit/s, ISO 7816, LIN, IrDA, modem control), Up to 3 SPIs (37.5 Mbits/s)

- 2 C_CAN interfaces (2.0B Active), SDIO interface
- USB 2.0 FS device/host/OTG controller with on-chip PHY, USB 2.0 HS/FS device/host/OTG controller with dedicated DMA,on-chip FS PHY and ULPI
- Hardware acceleration for AES 128, 192, 256, Triple DES, HASH (MD5, SHA-1)
- Analog true random number generator, CRC calculation unit, 96-bit unique ID
- RTC: subsecond accuracy, hardware calendar

Para más información de este chip, consultar el "datasheet".

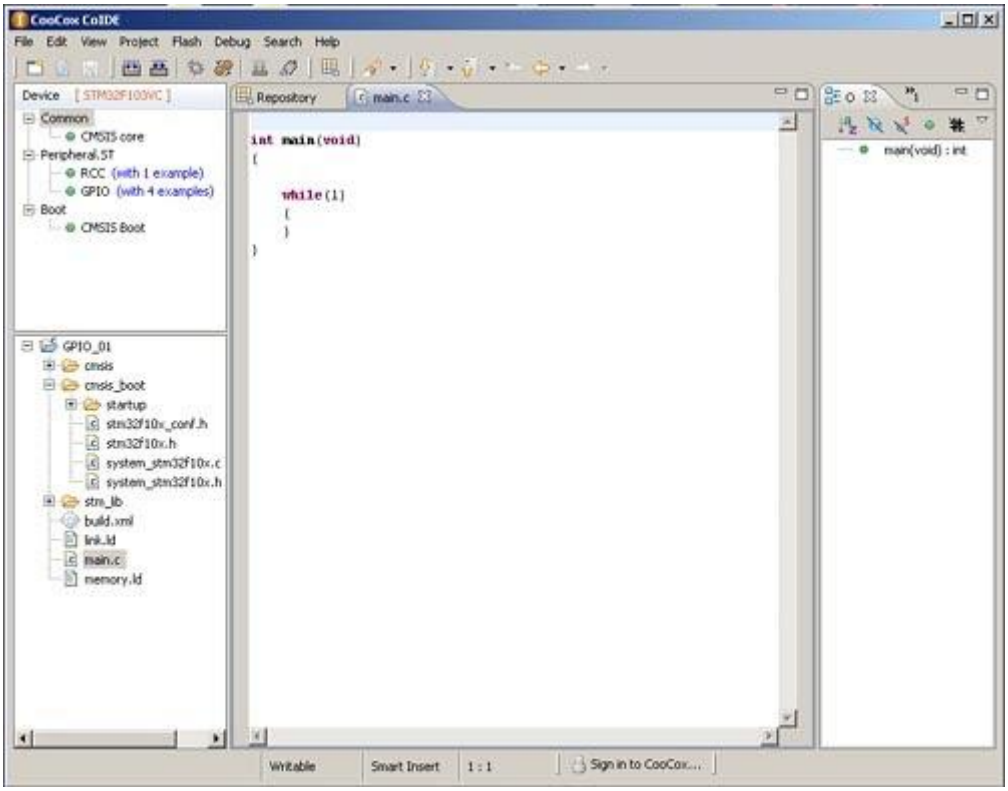
Una vez seleccionado el modelo concreto de microcontrolador, en la ventana central nos aparecen las librerías compatibles con el mismo, sin que tengamos que ir a buscarlas en largas listas sin saber muy bien que necesitamos para complementarlas. Únicamente dándole un doble click con el puntero sobre la línea correspondiente, nos mostrará en la ventana de la derecha las funciones contenidas en esta librería. Esta información es de gran ayuda, ya que no sólo muestra un listado y una corta descripción, sino que además, pulsando sobre los enlaces de los nombres, nos aparece la función concreta con su sintaxis y los parámetros que acepta. Otro detalle interesante de este programa es que al seleccionar una librería selecciona también de forma automática otras dependencias necesarias.



Al seleccionar las librerías, si no tenemos ningún proyecto abierto, el programa crea uno de forma automática y nos pedirá el nombre del mismo. Las carpetas que lo componen y que más tarde utilizaremos aparecerán en el recuadro inferior izquierda.

Las librerías que deberemos de insertar son las que componen el código expuesto en la memoria.

Una vez insertadas, en el recuadro inferior de la parte izquierda, veremos el archivo main.c que pulsando sobre él, si el proyecto es nuevo se nos abrirá una ventana que salvo dos escuetas órdenes el resto está vacío. Pues bien, aquí podemos empezar a crear nuestro programa.



Una vez desarrollado el programa, en caso de haber errores, el propio CooCox nos lo indicará con un interrogante en el margen izquierdo de la línea afectada. Se comprobará que no hay problemas y para iniciar la compilación pulsamos el cuarto icono de la barra superior o el "Built" de la pestaña "Project".

En la parte baja se abre una nueva ventana que nos va informando de las incidencias de la compilación, la cual de realizarse sin incidencias, acabará con un mensaje del tipo:

Program Size:

BUILD SUCCESFULL

Montaje del circuito de pruebas

Para el cableado del circuito, habrá que tener en cuenta los pines configurados en la programación. Los pines que se han utilizado en nuestro montaje son:

Pines de alimentación de la tarjeta

- VDD (Irá conectado a la salida del regulador de tensión L7805CV)
- GND (Conexión a masa)

Pines de alimentación de los sensores

- 3V (Esta será la alimentación del circuito de los sensores)
- GND (Conexión de masa del circuito de los sensores)

Pines del convertidor ADC

- PC 0 (Irá conectado a la señal del sensor/potenciometro)
- PC 1 (Irá conectado a la señal del sensor/potenciometro)
- PC 3 (Irá conectado a la señal del sensor/potenciometro)

Pines del Modulo de RF

- PA 10 (Se conectará a la entrada de Tx del receptor/transmisor)
- PB 6 (Se conectará a la entrada de Rx del receptor/transmisor)
- 3V (Se conectará a la entrada VCC del receptor/transmisor)
- GND (se conectará a la entrada GND del receptor/conector)

5.3 Control de calidad

Una vez finalizado el montaje, se verificará mediante una inspección visual que se hayan cumplido todas las exigencias en el montaje. Se hará hincapié en el mal conexionado de los pines, asegurando su correcta numeración. Se comprobará el estado de todos los cables asegurando que no hayan interconexiones entre pines.

5.4 Pruebas y ajustes finales

Una vez finalizados todos los apartados anteriores, encontrándose la placa montada completamente, y antes de la instalación de la misma se procederá a los ajustes necesarios para el buen funcionamiento del dispositivo.

Documento 3

Presupuesto

Índice

1. Introducción 3
2. Coste de los materiales 4
3. Coste de la mano de obra 6
3.1 Cuadro de precios elementales 6
3.2 Cuadro de tiempo empleado en la fabricación 6
3.3 Cuadro de formación de precios 7
3.4 Coste total de la mano de obra directa 7
4. Beneficio industrial 8
5. Coste total del proyecto 9

PRESUPUESTO

1. Introducción

En el presente documento se detalla el presupuesto del proyecto, el cual tiene como objetivo el proporcionar la información detallada sobre el coste global del mismo.

Con este fin, será desglosado en diversos apartados referentes a los diferentes conceptos que lo componen y que conjuntamente nos proporcionará el importe total del proyecto realizado.

Se debe considerar que cada presupuesto viene condicionado por múltiples y variados factores que intervienen en su desarrollo. Por ello antes de iniciar el estudio de este documento, se establecerán una serie de condiciones que se tendrán en cuenta en su evolución.

También se debe señalar, que el presupuesto que a continuación se desarrolla corresponde al coste del prototipo diseñado y montado. Además los tiempos de ejecución de cada una de las tareas necesarias para su desarrollo e implementación no se solaparán, por lo que el coste total del mismo será elevado; esto no quiere decir que, el producto final que se desease producir conllevase el coste que en este documento se detalla, puesto que como ya se ha dicho, se describe el presupuesto de desarrollo del prototipo.

Tendremos pues en cuenta las siguientes consideraciones:

- La elección del circuito y componentes del montaje correrá a cargo del ingeniero que firma el proyecto.
- Los precios de los componentes utilizados corresponden al precio de venta al público en establecimientos comerciales.

2. Coste de los materiales

Referencia	Descripción	Cantidad	Precio/ud	Importe
C1	condensador 100nf	1	0,06 €	0,06 €
C2	condensador 10nf	1	0,07 €	0,07 €
L7805CV	reguador tensión	1	0,71 €	0,71 €
Pot 100K	potenciometro multivuelta 100k	3	3,09 €	9,27 €
stm32f4	Kit microcontrolador	1	14,63 €	14,63 €
HM-TRP	Emisor/Receptor	1	18,20 €	18,20 €
RS 232 DB9	cable Usb serial converter	1	12,69 €	12,69 €
Mini USB	cable transmisión datos	1	5,72 €	5,72 €
Pc Acer 5742G	PC Portatil	1	699,00 €	699,00 €
Bosch 12V 80Ah	Bateria Bosch S6 12V 80Ah	1	211,27 €	211,27 €
Dupont 2,54mm	cables flexibles conector 2,54mm	20	0,23 €	4,60 €
SCA	Sensor caudal aire	1	37,24 €	37,24 €
STM	Sensor temperatura motor	1	13,85 €	13,85 €
TPS	Sensor posición acelerador	1	285,15 €	285,15 €
		subtotal con potenciometros		976,22 €
		subtotal con sensores		1.303,19 €

Como sabemos, también se han de presupuestar los medios auxiliares (también se conocen como imprevistos), y estos se suelen incluir en el presupuesto como el 2% del coste de materiales, de esta forma los materiales tendrán un coste de:

$$MEDIOS\ AUXILIARES = TOTAL\ MATERIALES \times 2\%$$

$$MEDIOS\ AUXILIARES = \ 1303{,}09 \times 0.02 = 26.0638€$$

MEDIOS AUXILIARES……26.06€

$$COSTE\ TOTAL\ MATERIALES = COSTE\ TOTAL + MEDIOS\ AUXILIARES$$

$$COSTE\ TOTAL\ MATERIALES = 1303{,}19 + 26{,}06 = 1329{,}25€$$

COSTE TOTAL MATERIALES……..1329,25€

3. Coste de la mano de obra

3.1 Cuadro de precios elementales

El coste de cada una de las categorías profesionales viene expresado mediante la tabla que se adjunta a continuación:

CATEGORÍA PROFESIONAL	COSTE (€/Hora)
Graduado en Ingeniería	14,00 €
Técnico especialista	9,00 €

3.2 Cuadro de tiempo empleado en la fabricación

El tiempo que necesitan emplear los distintos trabajadores en cada una de las fases de diseño, fabricación y montaje son aproximados, ya que en estos intervienen una serie de factores que podrían modificar a la alza o a la baja el tiempo real empleado por cada trabajador.

TAREA	GRADUADO EN INGENIERÍA	TÉCNICO ESPECIALISTA
Recogida de información y diseño	120 h	
Programación del microcontrolador		3 h
Montaje de los componentes		1 h
Configuración de la aplicación en el PC		2 h
Montaje y acondicionamiento del sistema		1h
Cableado e interconexión		1h
Inspección y ajuste	1 h	
TOTAL HORAS	**121 h**	**8 h**

3.3 Cuadro de formación de precios

UNIDAD	OPERARIO	CANTIDAD (H)	PRECIO UNITARIO (€)	TOTAL (€)
H	Gaduado Ingeniería	121	14,00 €	1.694,00 €
H	Técnico especialista	8	9,00 €	72,00 €
		SUBTOTAL		**1.766,00 €**

3.4 Coste total de la mano de obra directa

Al igual que con los materiales, en este apartado también se añadirán los medios auxiliares, que representaran el 2% del coste total de la mano de obra:

MEDIOS AUXILIARES = COSTE MANO DE OBRA * 2%

MEDIOS AUXILIARES = 1766,00 * 0,02 = 35,32 €

MEDIOS AUXILIARES …… 35,32 €

TOTAL M.O.D. = COSTE M.O.D. + MEDIOS AUXILIARES

TOTAL MANO DE OBRA DIRECTA = 1766,00 + 35,32 = 1801,32 €

TOTAL MANO DE OBRA DIRECTA …… 1801,32 €

4. Beneficio industrial

Para todo trabajo que se realice, toda empresa busca un beneficio industrial, éste se calcula mediante el producto del coste total de fabricación por un porcentaje determinado. Este porcentaje es arbitrario y depende de multitud de factores. El valor de este porcentaje suele variar entre un 20% y un 40%. En el cado de este proyecto se he considerado un 30%.

Para calcular el beneficio industrial, primero hay que saber el coste total de fabricación del producto:

COSTE TOTAL MATERIALES	976,22 €
COSTE TOTAL M.O.D	1.766,00 €
COSTE TOTAL DE FABRICACIÓN	**2.742,22 €**

Conocido el coste total de fabricación, se calcula el beneficio industrial:

BENEFICIO INDUSTRIAL = COSTE TOTAL DE FABRICACIÓN * 30%
BENEFICIO INDUSTRIAL = 2742,22 * 0'3 = 822,67 €

BENEFICIO INDUSTRIAL …… 822,67 €

5. Coste total del proyecto

Para el cálculo del coste total del proyecto, se suma el coste total de fabricación del producto y el beneficio industrial. Una vez realizado esto, se le sumará un 21% correspondiente al IVA, al valor resultante para la obtención del coste total:

COSTE TOTAL DE FABRICACIÓN	2.742,22 €
BENEFICIO INDUSTRIAL	822,67 €
COSTE SIN IVA	**3.564,89 €**

A este coste sin IVA se le añadirá el 21% correspondiente al IVA con lo que obtendremos el coste total del proyecto:

COSTE TOTAL DEL PROYECTO = COSTE SIN IVA + (COSTE SIN IVA * 21%)
COSTE TOTAL DEL PROYECTO = 3564,89 * 1’21 = 4313,51€

COSTE TOTAL DEL PROYECTO …… 4313,51 €

El prepuesto global del presente proyecto asciende a 4313,51 €

(CUATRO MIL TRES CIENTOS TRECE EUROS CON CINCUENTA Y UN CÉNTIMOS)