

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

SEDE CONCEPCIÓN"REY BALDUINO DE BELGICA"

Envió de datos por señal RF

TRABAJO PARA OPTAR AL TÍTULO PROFESIONAL DE

INGENIERO DE EJECUCIÓN EN CONTROL E INSTRUMENTACIÓN INDUSTRIAL

Alumno : Luis Fernández Villagrán

Profesor Guía : Felipe Benavidez

2018

Resumen

Debido a que hoy en día es muy necesario el obtener el dato de una variable y poder monitorearla de forma de remota, sin tener que ir directamente al lugar, previniendo accidentes, teniendo una respuesta más rápida y una comodidad de supervisión. Es por ello que hoy en día muchos equipos de medición tiene la factibilidad de transmitir datos por protocolos RS232, Ethernet, fibra óptica, etc. Facilitando él envió de datos por medios de cable o inalámbricos. Pero no muchas veces es tan factible realizar este tipo de instalaciones, conexiones y comunicaciones, ya sea por altos costo, o porque el lugar geográfico es muy complejo, y no permite que sea factible. Pues para ello nosotros hemos querido incorporar, aprovechando el avance de tecnología en los equipos de radiofrecuencia, en implementar este nuevo medio de comunicación, el cual puede abarcar estos lugares en que la fibra óptica, Ethernet, puntos de red, etc. No pueden ser aplicados. Pues en que consiste, es en usar un equipo de radio que puede abarcar distancias de 50km o más kms, dependiendo el tipo de instalación, y aprovechar su tecnología de envió de datos por señal RF, la cual se hace recepción por otro equipo , logrando la captura del dato y posteriormente la muestra de este valor en pantalla. Estos equipos no son de gran tamaño y solo necesitan que un equipo sea transmisión y otro receptor y ocupan como medio la señal Rf , la cual es transmitida por medio del aire o atmosfera.

ÍNDICE

Capítulo 1 1
1. Introducción 1
1.1 Descripción del Proyecto 6
1.2 Marco teórico 7
1.3 Objetivo general 8
1.4 Objetivos específicos 8
Capítulo 2 9
2. Especificaciones técnicas 9
2.1 Equipos de radio MOTOROLA DGM 8500 9
2.2 Accesorios de radio DGM8500 12
2.3 Arduinos (Se utilizará arduino mega y arduino uno): 17
2.4 Sensor DHT22 20
2.5 Cable RS232 en configuración NULL: 21
2.6 DT500 21
Capítulo 3 23
3. Software de programación 23
3.1Software MOTOTRBO CPS 15.0 23
Capítulo 4 26
4. Programación Arduino 26
4.1 Desarrollo de programación de arduino UNO como equipo de medición y envió de datos 26
4.2 Desarrollo de programación de arduino Mega 256 como equipo de recepción de variable y de mostrar valores en pantalla 30
Capítulo 5 33
5. Configuración de equipos de radio. 33
Capítulo 6 38
6.Configuracion del DT500 38
Capítulo 7 42
7. Puesta en marcha 42
Conclusión 47
Resultados obtenidos: 47
Bibliografía 48
Anexos 49

Tabla de valores para arduino receptor:.....49
Programación Arduino transmisor:.....52

Capítulo 1

1. Introducción

Antes de comenzar se hará mención de que es una señal Rf, porque medio se transmite y que equipos se ocupan para ello.

Una señal Rf, es una señal de radio frecuencia el cual es un conjunto de frecuencias situado en los rango de 3Hz hasta los 300Ghz, siendo una energía del espectro electromagnético baja.

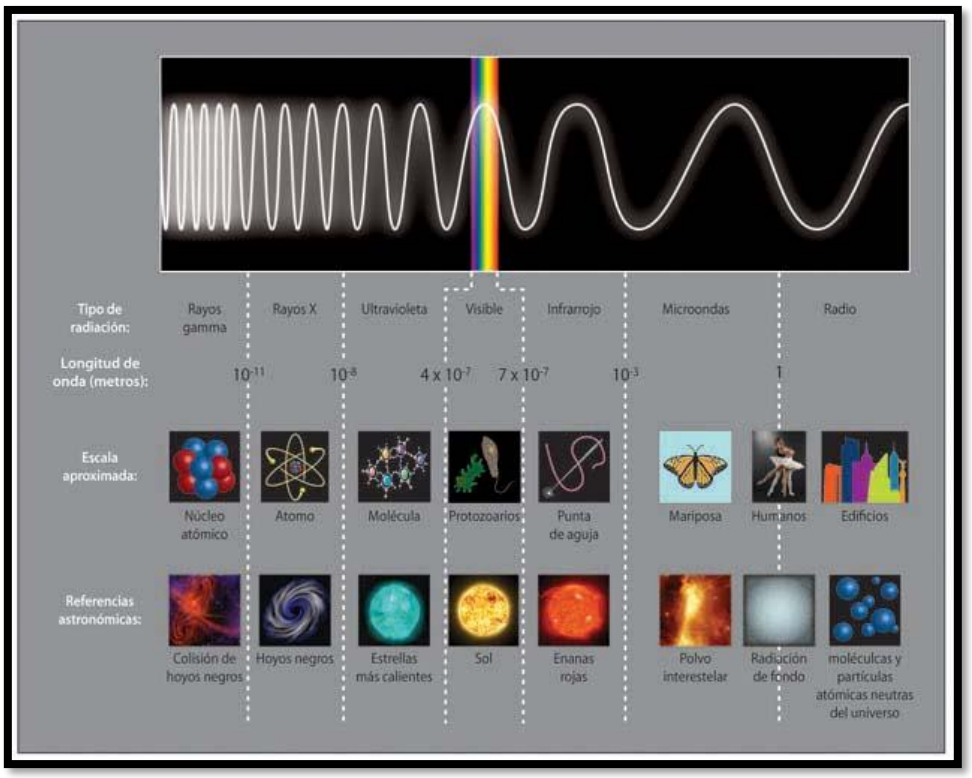


Figura 1.1 Banda de frecuencia

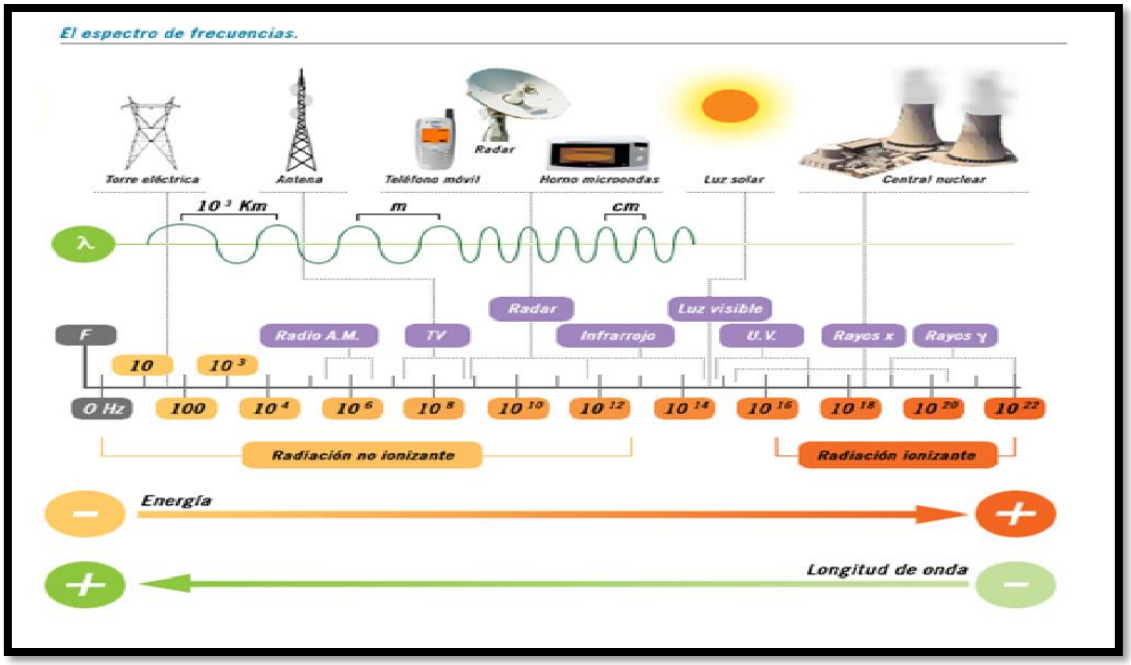


Figura 1.2 Espectro frecuencia

La frecuencia es una magnitud que mide el número de veces que una señal se repite en una unidad de tiempo. Se mide en Hz. Y la longitud de onda es la distancia que una señal recorre en el intervalo de tiempo comprendido entre dos máximos consecutivos y que es inversamente proporcional a la frecuencia de la señal. Se mide en lambda.

Dentro de la bandas de radio frecuencias encontramos el siguiente rango según frecuencia y longitud de onda.

Frecuencia	Siglas	Intervalo de frecuencia	Longitud de onda
Extra baja	ELF	3 - 30 Hz	10^5 - 10^4 km
Súper baja	SLF	30 - 300 Hz	10^4 - 10^3 km
Ultra baja	ULF	300 - 3000 Hz	10^3 – 10^2 km
Muy baja	VLF	3 - 30 kHz	10^2 - 10 km
Baja	LF	30 - 300 kHz	10 - 1 km
Media	MF	300 - 3000 kHz	1 – 0.1 km
Alta	HF	3 - 30 MHz	100 - 10 m
Muy alta	VHF	30 - 300 MHz	10 - 1 m
Ultra alta	UHF	300 - 3000 MHz	1 – 0.100 m
Súper alta	SHF	3 - 30 GHz	100 - 10 mm
Extra alta	EHF	30 - 300 GHz	10 - 1 mm

Figura 1.3 Intervalo de frecuencia

Las frecuencias de ondas de radio son usadas para la televisión, emisiones de radio FM y AM, comunicaciones militares, teléfonos celulares, radioaficionados, redes inalámbricas de computadoras, y otras numerosas aplicaciones de comunicaciones.

La mayoría de las ondas de radio pasan libremente a través de la atmósfera de la Tierra. Sin embargo, algunas frecuencias pueden ser reflejadas o absorbidas por las partículas cargadas de la ionosfera.

Las ondas de radio mientras se propagan pueden sufrir refracción, reflexión, dispersión y difracción:

Refracción: La seña sufre un cambio de trayectoria al chocar con objetos de distinta densidad.

Reflexión: la onda de radio puede atravesar las distintas capas de la atmosfera. De la troposfera hasta la ionosfera, y en algunos casos se produce reflexión total.

Dispersión: Sucede cuando la señal atraviesa algunas masas de electrones o pequeñas capas de agua, en espacios suficientemente grandes.

Difracción: Es el esparcimiento de la señal en los límites de una superficie. Para ello tiene que haber un obstáculo, y esto permite que la seña llegue al otro lado del objeto.

Estos fenómenos que se han mencionados permiten la comunicación a largas distancias, pudiendo existir distintos tipos de propagación como:

Propagación por onda directa: Para este tipo de propagación debe haber una línea vista entre emisor y receptor. Se ocupan señales por encima de los 50 MHz, ya que las frecuencias altas se ven menos afectadas por los fenómenos atmosféricos y no requieren de grandes antenas para tener una transmisión efectiva.

Propagación por onda terrestre: Es posible gracias a la difracción, permitiendo que la señal Rf viaje por la curvatura de la tierra, y así esta alcance largas distancias antes de ser absorbida por la tierra.

Propagación por onda refractada o ionosférica: Es muy importante este tipo de propagación. Aquí actúa la atmosfera como reflector y a su vez en la ionosfera.

Propagación troposférica: En esta capa se forman las nubes y la temperatura desciende rápidamente, debido a la altura donde se producen estos cambios de temperatura, se producen los canales de ionización, y estos canales son ideales para que las señales de Rf puedan viajar. Es ideal para frecuencias VHF y UHF.

Existen más formas de propagación pero solo mencionaremos estás.

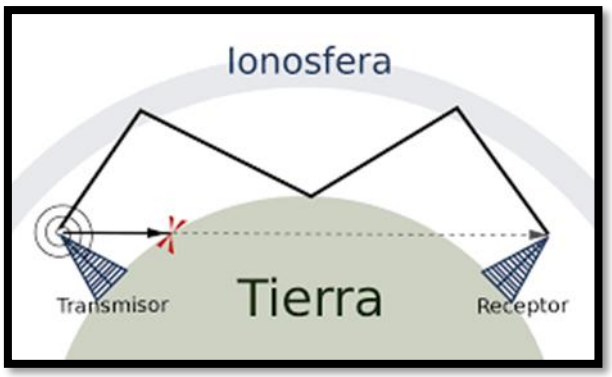


Figura 1.4 Transmisión por ionosfera

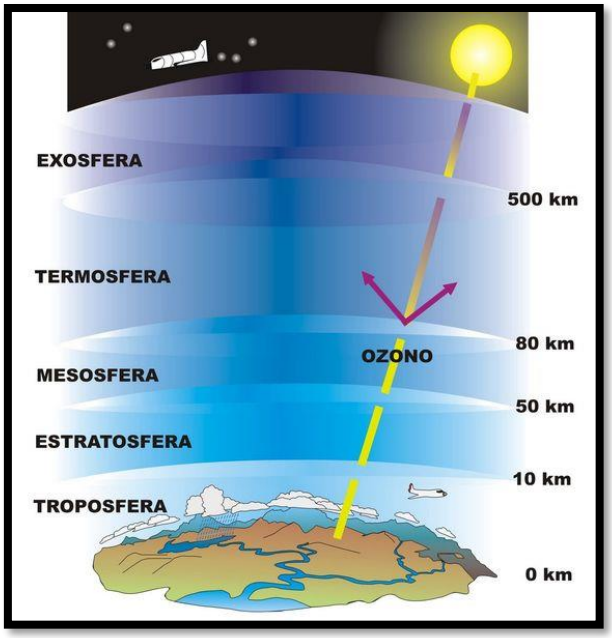


Figura 1.5 Capas de la atmosfera

Un radiotransmisor es un dispositivo electrónico que, mediante una antena, irradia ondas electromagnéticas que contienen información, como ocurre en el caso de las señales de radio, televisión, telefonía móvil o cualquier otro tipo de radiocomunicación. En las comunicaciones un transmisor es un equipo que emite una señal, código o mensaje a través de un medio. Para lograr una sesión de comunicación se requiere: un transmisor, un medio y un receptor. El transmisor tiene como función codificar señales ópticas, mecánicas o eléctricas, amplificarlas, y emitirlas como ondas electromagnéticas a través de una antena. La codificación elegida se llama modulación. Esta viaja por el medio hasta llegar al receptor el cual la decodifica.



Figura 1.6 Medios de transmisión

Con lo que ya se ha mencionado, se entenderá porque medio queremos hacer esta comunicación, entre la unidad de medición y la unidad que mostrara en pantalla el resultado, y se hablara ahora en el proyecto en sí.

1.1 Descripción del Proyecto

El proyecto busca el poder realizar envió de datos a través de equipos de radio comunicación profesional, los cuales son usados a nivel industrial y militar. Y lograr hacer uso de esta nueva tecnología que Motorola y otras emperezas han implementado en sus equipos, llevando el uso de radio no solo a la transmisión de voz, sino también a la transmisión de datos, procurando abarcar un campo comunicación entre personas (por voz) y entre equipo y equipo (dato) . Para ello se hará uso de radios Motorola las cuales son radios que permiten comunicación de voz en señal análoga y señal digital. Ocupando esta última para la incorporación de él envió de datos. Pero para ello también se hará uso del dispositivo DT500, el cual hace el enlace entre la unidad de medición y la radio (transmisor), y entre la radio (receptor) y la unidad de visualización. Como equipo de medición se usara dos placas arduino. Un arduino uno y un arduino Mega, los cuales permiten una programación más didáctica y fácil de adaptar a las radios y contempla accesorios como uso de sensores y pantallas, de los cuales se usaran.

1.2 Marco teórico

Para dar solución al envió datos en lugares de largas distancias, sin posibilidad de cablear por costos altos o geográficos, se utilizará las señales RF, como transporte de nuestros datos. Para ello se hará uso de radios de comunicación en UHF con las cuales se harán él envió de datos, teniendo que programar y seleccionar un canal, el cual se hará con un cifrado para evitar interrupciones en él envió de datos. Pero para poder capturar la variable , usaremos la interfaz DT500, la cual da el enlace entre el sensor, equipo de medición o PLC y la radio por comunicación RS232, permitiéndo establecer comunicación entre la variable y la radio, y comunicación entre la radio y la pantalla de monitoreo o PLC. En este caso el equipo de medición será el arduino el cual tendrá incorporado un sensor de temperatura y humedad. El arduino será el encargado de leer la variable del sensor, tratarla y enviarla por el DT500 a la radio y por el otro lado, otro arduino será el encargado de recibir el dato que capture la segunda radio, y codificarla y mostrarla en un pantalla LCD para arduino.

1.3 Objetivo general

Implementar el uso de envió de datos a través de señal RF de largo alcance.

1.4 Objetivos específicos

- Configuración y programación de radio DGM 8500
- Configuración de dispositivo DT500
- Creación de programa de los dos Arduino (uno y mega 2560)
- Instalación de las dos radios Motorola DGM 8500 con su accesorios
- Conexión de los equipos de radio, arduino y dispositivo DT500
- Puesta en marcha de la comunicación de dato por señal de RF

Capítulo 2

2. Especificaciones técnicas

En el siguiente capítulo se detallan las principales características de los equipos utilizados en el proyecto, imágenes, datasheet, conexionados, configuraciones etc.

2.1 Equipos de radio MOTOROLA DGM 8500

Este dispositivo móvil con todas las funciones transforma su empresa al ofrecer comunicaciones de voz y datos sin igual, con audio y datos Bluetooth integrados, GPS integrado, mensajería de texto y una pantalla de 4 líneas a todo color con modo diurno y nocturno para que la lectura de órdenes de trabajo y mensajes de texto sea más fácil. También obtendrá un audio de la mejor calidad, que incluye audio inteligente y una función personalizable de anuncios de voz.

HOJA DE ESPECIFICACIONES DE PRODUCTO
RADIOS MÓVILES SERIE DGM™8000 / DGM™5000 MOTOTRBO

ESPECIFICACIONES DE LAS SERIES DGM™8000 Y DGM™5000

TOTALMENTE COMPATIBLES CON LOS RADIOS LEGADOS MOTOTRBO.

ESPECIFICACIONES GENERALES

		DGM™8500/DGM™5500			DGM™8500	DGM™8000/DGM™5000			DGM™8000
		VHF	**UHF Banda 1**	**UHF Banda 2**	**800/900**	**VHF**	**UHF Banda 1**	**UHF Banda 2**	**800/900**
Capacidad de canal		Hasta 1.000			Hasta 1.000	99			99
Salida RF típica	Baja potencia	1-25 W	1-25 W			1-25 W	1-25 W		
	Alta potencia	25-45 W	25-40 W	1-40 W	806-870 MHz: 10-35 W 896-941 MHz: 10-30 W	25-45 W	25-40 W	1-40 W	806-870 MHz: 10-35 W 896-941 MHz: 10-30 W
Dimensiones (A X A X L)		2,1 x 6,9 x 8,1 in (53,3 x 175,3 x 205,7 mm)							
Peso		3,9 lbs (1,8 kg)							
Consumo de energía	Standby	0,81 A							
	Rx en audio nominal	2 A							
	Transmisión	1-25 W: 11,0 A 25-45 W: 14,5 A	1-25 W: 11,0 A 25-40 W: 14,5 A	1-25 W: 11,0 A 25-40 W: 14,5 A	12 A	1-25 W: 11,0 A 25-45 W: 14,5 A	1-25 W: 11,0 A 25-40 W: 14,5 A	1-25 W: 11,0 A 25-40 W: 14,5 A	12A
Descripción de la FCC		1-25 W: ABZ99FT3086	1-25 W: ABZ99FT4087	1-40 W: ABZ99FT4085	AZ492FT5862	1-25 W: ABZ99FT3086	1-25 W: ABZ99FT4087	1-40 W: ABZ99FT4085	AZ492FT5862
		25-45 W: ABZ99FT3087	25-40 W: ABZ99FT4088			25-45 W: ABZ99FT3087	25-40 W: ABZ99FT4088		
Descripción del IC		1-25 W: 109AB-99FT3086	1-25 W: 109AB-99FT4087	1-40 W: 109AB-99FT4085	109U-92FT5862	1-25 W: 109AB-99FT3086	1-25 W: 109AB-99FT4087	1-40 W: 109AB-99FT4085	109U-92FT5862
		00 25-45 W: 109AB-99FT3087 0	25-40 W: 109AB-99FT4088			25-45 W: 109AB-99FT3087	25-40 W: 109AB-99FT4088		

RECEPTOR

	VHF	**UHF Banda 1**	**UHF Banda 2**	**800/900**
Frecuencias	136-174 MHz	403-470 MHz	450-512 MHz	806-870 MHz 896-941 MHz
Espaciamiento de canal	12,5 kHz / 25 kHz*			806-870 MHz 12,5/25 kHz* 896-941 MHz: 12,5 kHz
Estabilidad de frecuencia (-30°C, +60°C, +25°C Ref)	± 0,5 ppm			
Sensibilidad analógica (12dB SINAD)	0,3uV; 0,22uV (typical)			
Sensibilidad digital	5% BER @ 0,25uV (0,19uV typical)			5% BER @ 0,3uV
Intermodulación (TIA603D)	78 dB			75 dB
Selectividad de canal adyacente (TIA603D)	50 dB @ 12,5 kHz 80 dB @ 25 kHz*	50 dB @ 12,5 kHz 75 dB @ 25 kHz*		
Rechazo espúreo (TIA603D)	80 dB	75 dB		
Audio nominal	3 W (Interno) 7.5 W (Externo - 8 ohms) 13 W (Externo - 4 ohms)			
Distorsión de audio en audio nominal	3% (typical)			
Interferencia y ruido	-40 dB @ 12,5 kHz/-45 dB @ 25 kHz*			
Respuesta acústica	TIA603D			
Emisión espúrea conducida (TIA603D)	-57dBm			

TRANSMISOR

	VHF	**UHF Banda 1**	**UHF Banda 2**	**800/900**
Frecuencias	136-174 MHz	403-470 MHz	450-512 MHz	806-870 MHz 896-941 MHz
Espaciamiento de canal	12,5 kHz / 25 kHz*			806-870 MHz: 12,5/25 kHz* 896-941 MHz: 12,5 kHz
Estabilidad de frecuencia (Ref. -30°C, +60°C, +25°C)	± 0,5 ppm			
Baja potencia de salida	1-25 W			
Alta potencia de salida	25-45 W		1-40 W	806-870MHz 10-35W 896-941MHz 10-30W
Restricción de modulación	± 2,5 kHz @ 12,5 kHz/± 5,0 kHz @ 25 kHz*			
Interferencia y ruido en FM	-40 dB @ 12,5 kHz/-45 dB @ 25 kHz*			
Emisión conducida/radiada	-36 dBm < 1 GHz/-30 dBm > 1 GHz			
Potencia de canal adyacente	60 dB @ 12,5 kHz/70 dB @ 25 kHz*			50 dB @ 12,5 kHz/ 60 dB @ 25 kHz*
Respuesta acústica	TIA603D			
Distorsión de audio	3%			
Modulación FM	12,5 kHz: 11K0F3E / 25 kHz*: 16K0F3E			
Modulación digital 4FSK	12,5 kHz Data Only: 7K60F1D & 7K60FXD			
	12,5 kHz Voice: 7K60F1E & 7K60FXE			
	Combinación de voz y datos (12.5 kHz): 7K60F1W			
Tipo de vocodificador digital	AMBE+2™			
Protocolo digital	ETSI TS 102 361-1, -2, -3			

*25 kHz NO está disponible en los EE.UU. Reglas de la FCC narrowbanding no permiten el funcionamiento de este modelo, en configuración de 25kHz en la Parte 90 de frecuencias VHF / UHF

MOTOROLA

Figura 2.1 Datasheet Motorola

HOJA DE ESPECIFICACIONES DE PRODUCTO
RADIOS MÓVILES SERIE DGM™8000 / DGM™5000 MOTOTRBO

ESTÁNDARES MILITARES

	810C		810D		810E		810F		810G	
MIL-STD APLICABLE	MÉTODO	PROCEDIMIENTOS	MÉTODO	PROCEDIMIENTOS	MÉTODO	PROCEDIMIENTOS	MÉTODO	PROCEDIMIENTOS	MÉTODO	PROCEDIMIENTOS
Baja presión	500.1	I	500.2	II	500.3	II	500.4	II	500.5	II
Alta temperatura	501.1	I, II	501.2	I/A1,II/A1	501.3	I/A1,II/A1	501.4	I/Caliente, II/Caliente	501.5	I/A1, II
Baja temperatura	502.1	I	502.2	I/C3, II/C1	502.3	I/C3, II/C1	502.4	I/C3, II/C1	502.5	I/C3, II
Choque térmico	503.1	-	503.2	I/A1/C3	503.3	I/A1/C3	503.4	I	503.5	I/C
Radiación solar	505.1	II	505.2	I	505.3	I	505.4	I	505.5	I/A1
Lluvia	506.1	I, II	506.2	I, II	506.3	I, II	506.4	I, III	506.5	I, III
Humedad	507.1	II	507.2	II	507.3	II	507.4	-	507.5	II - Agravado
Niebla salina	509.1	-	509.2	-	509.3	-	509.4	-	509.5	-
Polvo	510.1	I	510.2	I	510.3	I	510.4	I	510.5	I
Vibración	514.2	VIII/F, Curva-W	514.3	I/10, II/3	514.4	I/10, II/3	514.5	I/24	514.6	I/24
Golpes	516.2	I, II	516.3	I, IV	516.4	I, IV	516.5	I, IV	516.6	I, IV, V, VI

GPS

Especificaciones sobre precisión para seguimiento a largo plazo (95° valor percentil >5 satélites visibles con una intensidad de señal nominal de -130 dBm).	
TTFF (Tiempo para el primer punto fijo) - Arranque en frío	< 1 minuto
TTFF (Tiempo para el primer punto fijo) - Arranque en caliente	< 10 segundos
Precisión horizontal	< 5 metros

BLUETOOTH

Versión	Admite Bluetooth® 2.1 + Especificación EDR
Perfiles admitidos	Perfil de Auriculares Bluetooth (HSP), Perfil de Puerto Serie (SPP), PTT rápido Motorola.
Dispositivos admitidos	El radio admite 1 accesorio de audio Bluetooth y 1 dispositivo de datos Bluetooth simultáneamente
Alcance	Clase 2; 10 metros

ESPECIFICACIONES AMBIENTALES

Temperatura de funcionamiento	-30°C - +60°C
Temperatura de almacenamiento	-40°C - +85°C
Choque térmico	Según MIL-STD
Humedad	Según MIL-STD
ESD	IEC 61000-4-2 Nivel 3
Ingreso de agua y polvo	IP54, MIL-STD
Prueba de embalaje	MIL-STD 810C, D, E, F y G

Especificaciones sujetas a cambios sin notificación previa. Todas las especificaciones incluidas en este documento son especificaciones típicas. El radio cumple con todos los requisitos reglamentarios vigentes.
Versión 1.07/11

Para más información acerca de cómo fortalecer sus operaciones de voz móvil, visite www.motorolasolutions.com/mototrbo

MOTOROLA, MOTO, MOTOROLA SOLUTIONS y el logotipo de la M estilizada son marcas comerciales o marcas comerciales registradas de Motorola Trademark Holdings, LLC y son utilizadas bajo licencia. Todas las demás marcas comerciales pertenecen a sus respectivos propietarios.
© 2014 Motorola Solutions, Inc. Todos los derechos reservados.
2014-06

MOTOTRBO
REINVENTANDO DIGITAL

MOTOROLA

Figura 2.2 Estándares

2.2 Accesorios de radio DGM8500

Con respecto a las antenas hay varios tipos , pero se hará mención de dos antenas. Siendo la primera la que se ocupará para las pruebas y demostraciones.

Antena móvil ¼ de onda UHF 470-512 MHZ:



Figura 2.3 Antena móvil

Base magnética para antena móvil con cable RG58:



Figura 2.4 Base magnética

Antena OPEK 7/8 UHF :

-Rango de Frecuencia: 400-500 MHz.

-Ganancia: 5 dB.

-Ancho de Banda: 10 MHz.

-Potencia: 200 W.

-Conector: UHF Hembra.

-Longitud Máxima: 1 m.

-Resistencia al viento: 193 Km/h

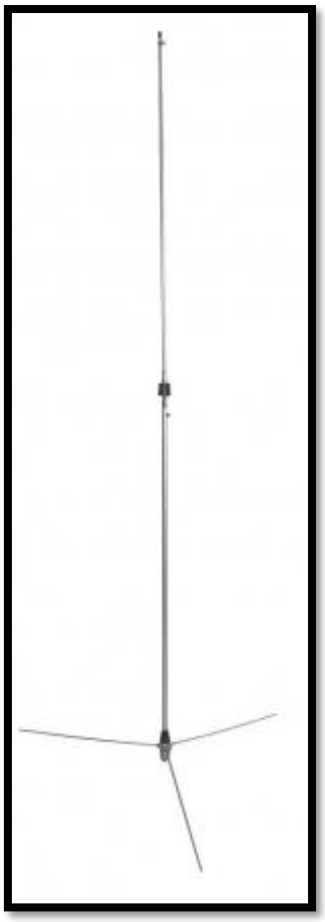


Figura 2.5 Antena base

Esta antena se instala sobre un brazo o mástil galvanizado y se utiliza un cable RG88 9913



Figura 2.6 Cable RG8



Figura 2.7 Cable RG58



Figura 2.8 Antena sobre mástil

Conectores:

Para RG58:



Figura 2.9 Mini u macho

PL259 para cable RG8 9913:



Figura 2.10 PL 259 macho

Fuente de poder 13.8 Vcc

Caracteristicas:

SEC-1223BBM SAMLEX FUENTE DE PODER SWITCHING (Conmutada) 23 Amp Con Reversion de baterías.

Voltaje: Input: -230VAC

Voltaje salida: 13,8 VCC (ajust.)

Corriente máxima 13,8V: 23 Amp. Contínuos y 25 Amp. intermitentes

Potencia máxima: 375W

Refrigeración automática (1v.)

Protección por cortocircuito, sobreintensidad, sobrevoltaje y sobrecalentamiento.

Rizado y RFI mínimos.

Medidas: 177 x 57 x 190 mm.

Peso aproximado: 1,5 Kg.



Figura 2.11 Fuente de poder

Cable de alimentación Motorola con fusible de 15 o 10 Amper:



Figura 2.12 Cable de alimentación Motorola

2.3 Arduinos (Se utilizará arduino mega y arduino uno):

Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos.

Arduino puede sentir el entorno mediante la recepción de entradas desde una variedad de sensores y puede afectar a su alrededor mediante el control de luces, motores y otros artefactos. El microcontrolador de la placa se programa usando el Arduino Programming Language (basado en Wiring) y el Arduino Development Environment(basado en Processing). Los proyectos de Arduino pueden ser autónomos o se pueden comunicar con software en ejecución en un ordenador (por ejemplo con Flash, Processing, MaxMSP, etc.).

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Figura 2.13 Datasheet arduino

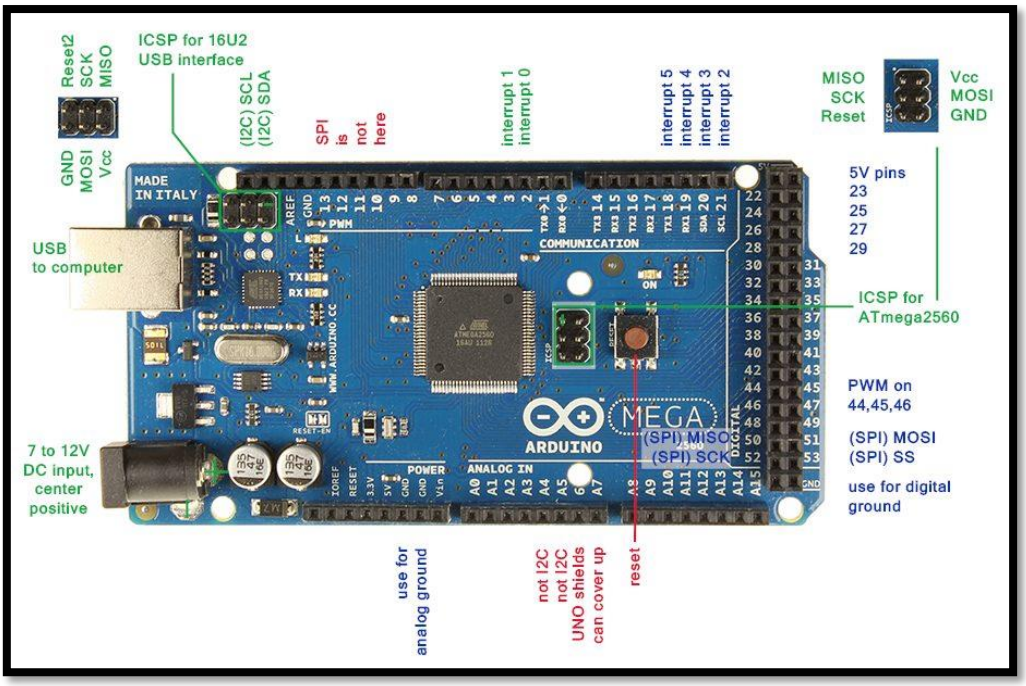


Figura 2.14 Arduino mega

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Figura 2.15 Datos técnicos

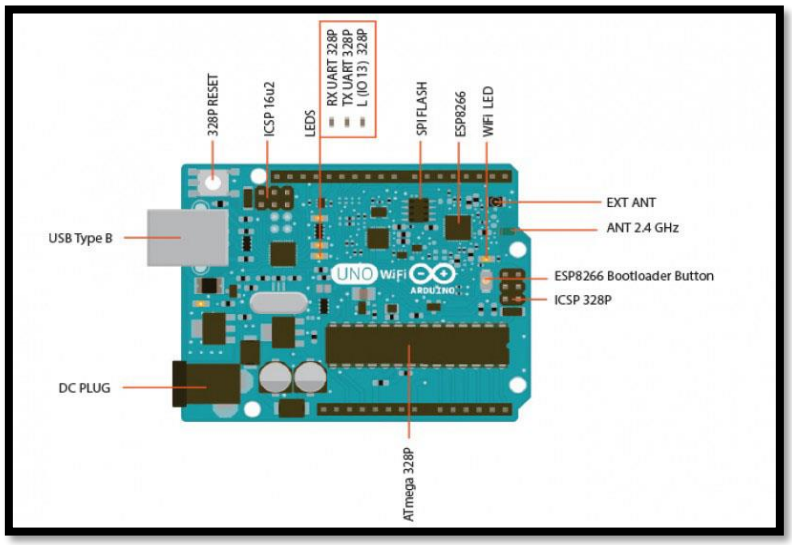


Figura 2.16 Arduino uno

2.4 Sensor DHT22

DHT22 es un sensor de humedad y temperatura ambiental, genera una señal digital calibrada de fábrica. Utiliza una técnica de colección de señales digitales y tecnología de medición de humedad el cual asegura su confiabilidad y estabilidad. Posee un micro controlador integrado de 8 bit el cual está conectado directo al sensor y envía los pulsos.

-Medición de temperatura entre -40 a 125, con una precisión de 0.5ºC

-Medición de humedad entre 0 a 100%, con precisión del 2-5%.

-Frecuencia de muestreo de 2 muestras por segundo (2 Hz)

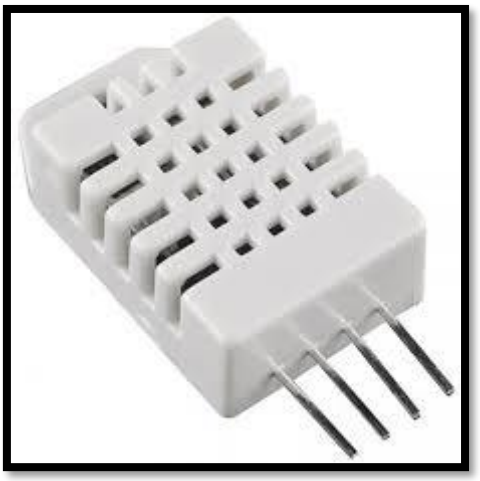


Figura 2.17 DHT 22

2.5 Cable RS232 en configuración NULL:



Figura 2.18 Cable RS232

2.6 DT500

El módem VoIP se ha diseñado para permitir que una amplia gama de dispositivos, como SCADA, AMR, CAN BUS, aplicaciones de telemetría, alimenten información específica en un sistema TRBOnet Dispatch.

Características destacadas

• Interfaz RS232

• 7 pines de E / S programables para telemetría y telemando

• Almacenamiento interno para GPS y eventos de radios

• Geofencing y alarmas

• Condiciones configurables y acciones

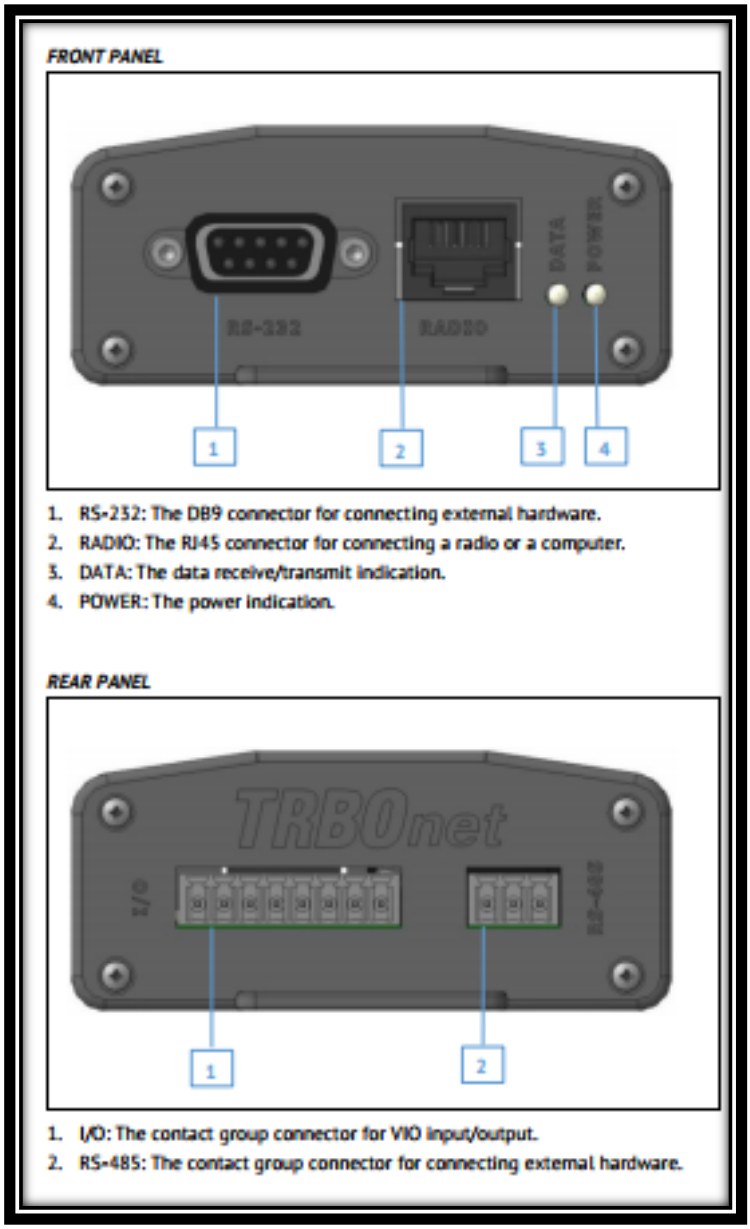


Figura 2.19 DT500

Capítulo 3

3. Software de programación

Brevemente mostrarán y comentarán un poco sobre los software de programación , que ocuparemos para los equipos de radio, DT500 y arduino.

3.1Software MOTOTRBO CPS 15.0

Este software permite acceder a varias configuraciones de la radio, desde crear un canal de comunicación (el cual ocuparemos para enviar dato), cifrar dicho canal, asignar un número de identificación de la radio, habilitar lectura de conector de accesorios, entre muchas otras funciones.

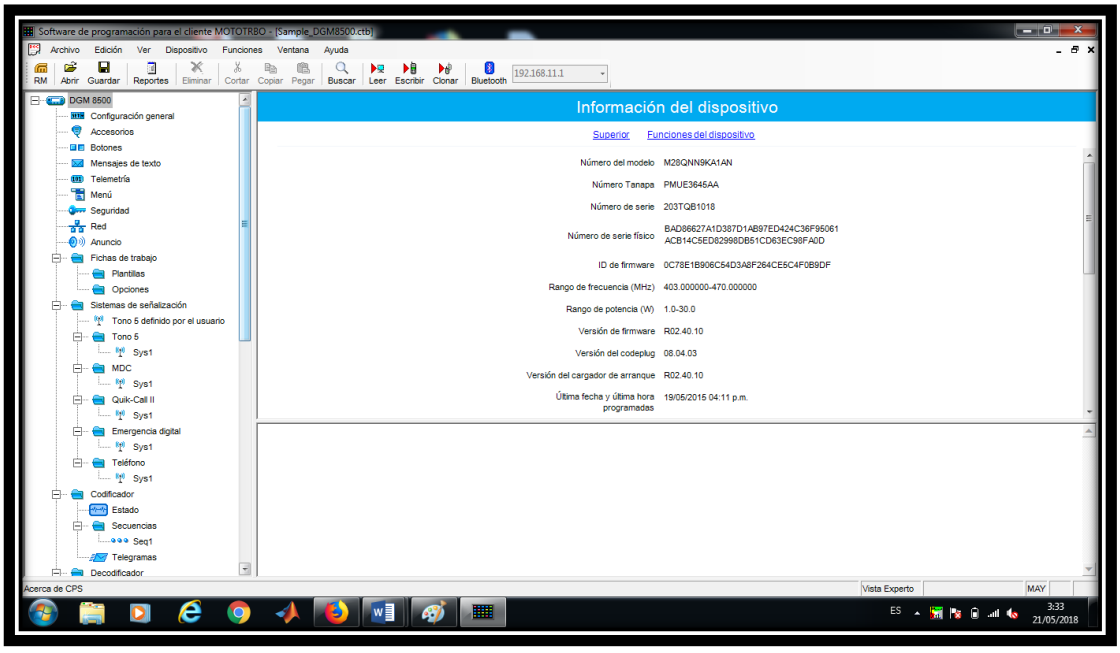


Figura 3.1 Software Motorola

3.2 Software TRBONET SWIFT

Programa que da el acceso a configurar el DT500 para el uso de sus distintas funciones, asignar ID, definir tipo de comunicación, etc.

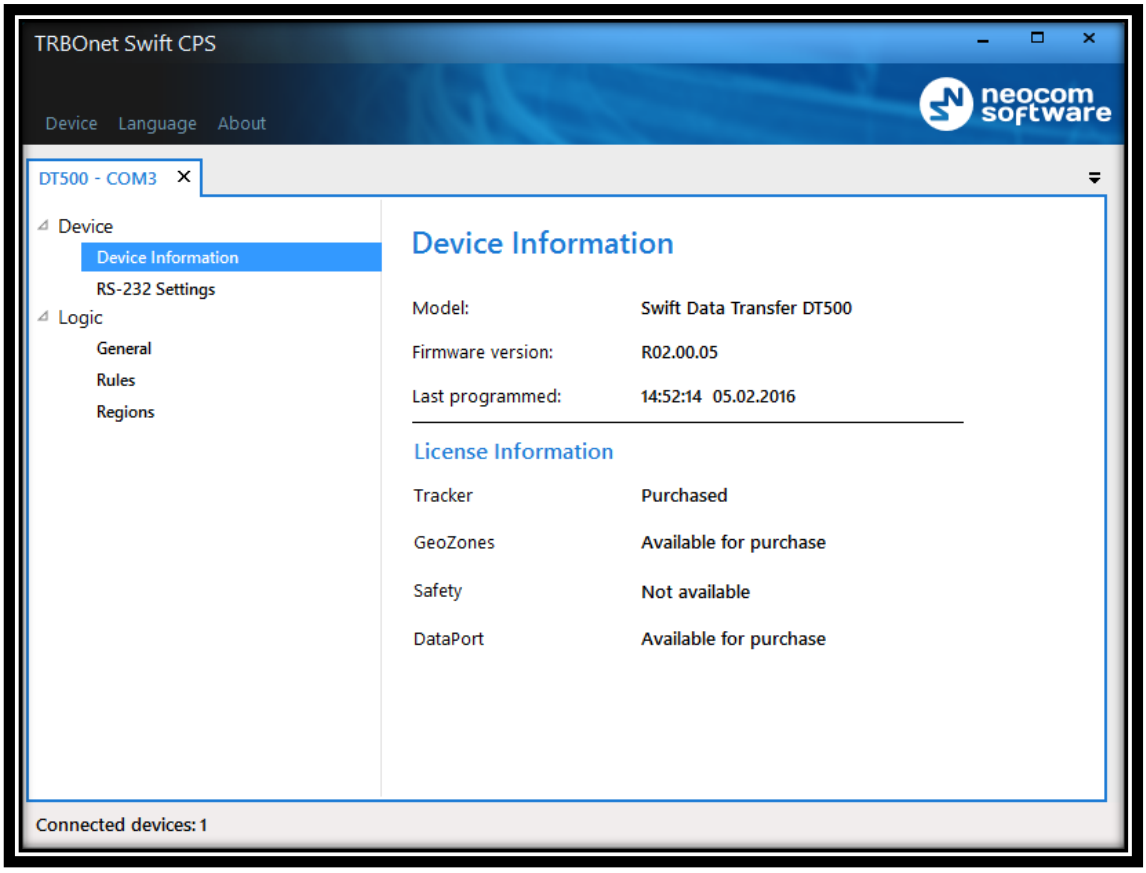


Figura 3.2 Software DT500

3.3 Software Arduino 1.8.5

Con esta plataforma se podrá crear múltiples programas, cargar librerías que nos permitirán ocupar los distintos accesorios y sensores que podemos conectar al arduino. Se podrá cargar un programa ya sea creado desde cero o desde uno ya creado y modificarlo. Este programa maneja un sinfín de modelos de arduino, e incluso se pueden descargar otros tipos de tarjeta arduino que no estén en la lista propia.

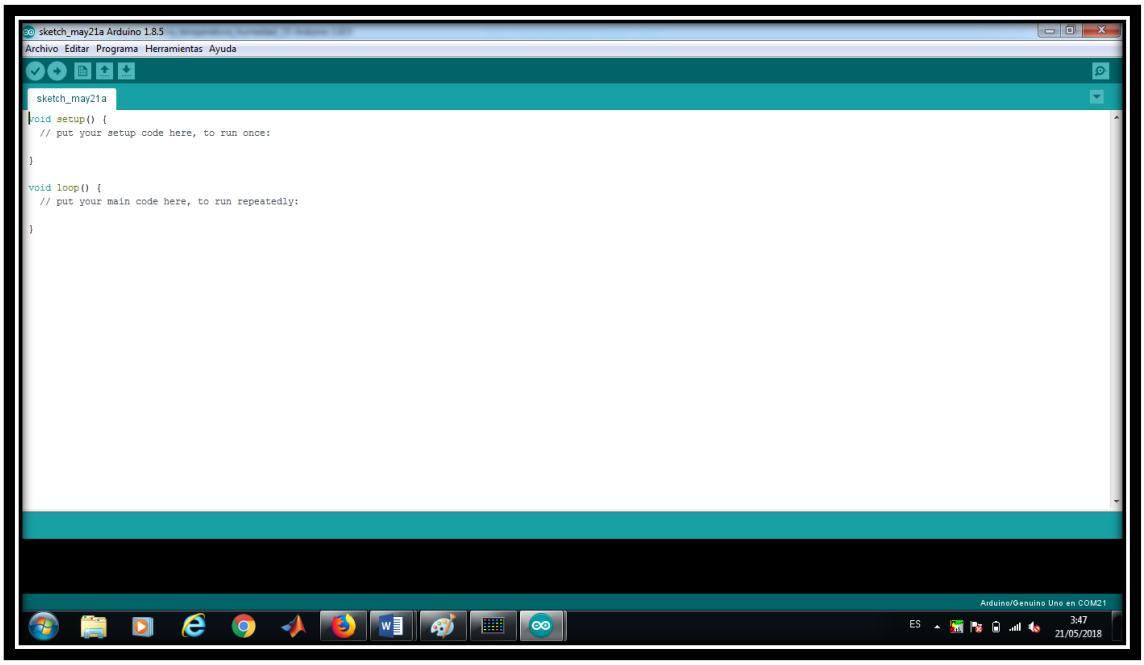


Figura 3.3 Software arduino

Capítulo 4

4. Programación Arduino

El desarrollar un proyecto bajo la plataforma arduino, resulta muy sencillo, debido a lo intuitivo y amigable que es el software.

4.1 Desarrollo de programación de arduino UNO como equipo de medición y envió de datos

A continuación se mostrará extractos de la programación del transmisor, esta no corresponde a la programación completa, solo aquellas parte de la estructura que son más relevantes y con su respectivo comentario.

En esta ventana se verá que incluimos la librería del sensor a ocupar, el cual es un sensor de humedad. También se define el pin a utilizar para la lectura y el modelo del sensor. Finalmente se inicia la comunicación del sensor con el arduino.

Por otra parte se definen nuestras variables a utilizar donde capturaremos el valor del sensor , tanto para temperatura y humedad, como también la variable que definirá al receptor que tipo de dato se le está enviando (humedad o temperatura).

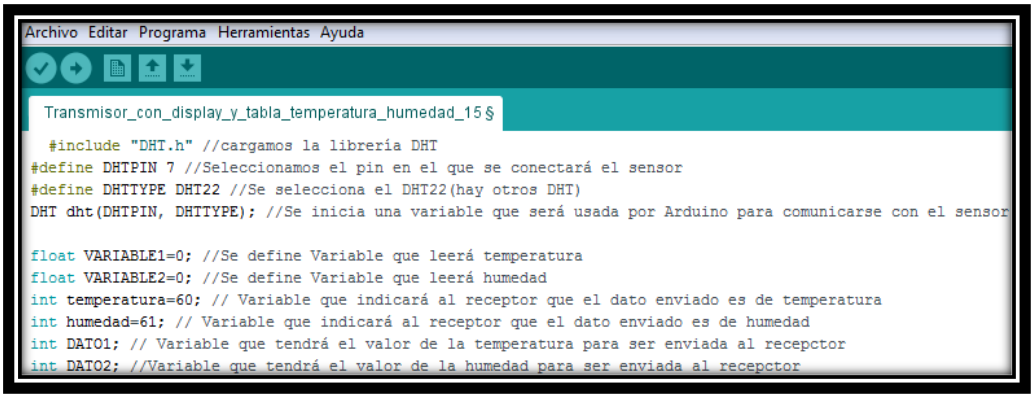
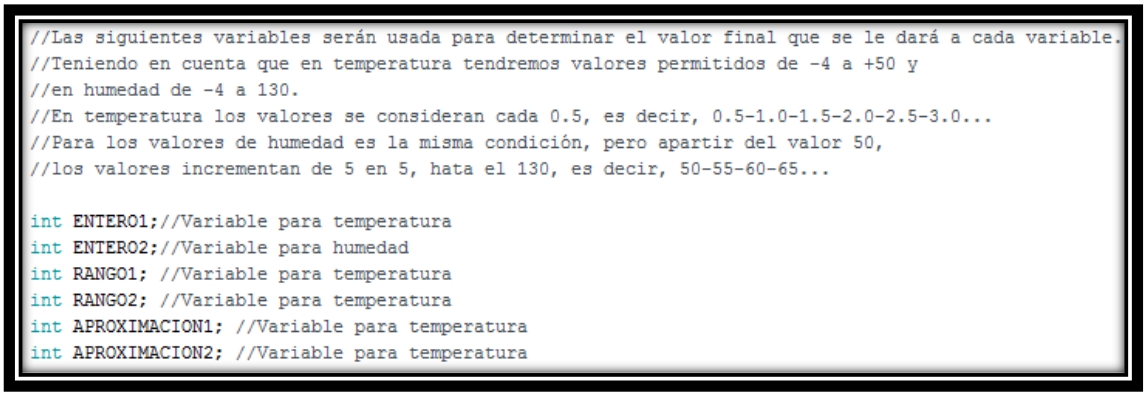


Figura 4.1 Programa arduino RX

En esta imagen se muestran las variables definidas que se utilizaran para emplear un cálculo matemático para ir tratando el dato leído y poder enviarlo en un formato entendible y codificable para el arduino receptor. Esto dado que al enviar un dato sin tratarlo, se produce un error, y el mismo valor toma distintos datos al ser recibidos por el receptor (es solo para este caso al usar arduino). Cabe señalar que con este tratamiento, se pueden manejar valores de temperatura entre -4.5 grado y 50 grados, con un intervalo de 0.5, y para humedad entre 0% y 130%, con un intervalo de 5.

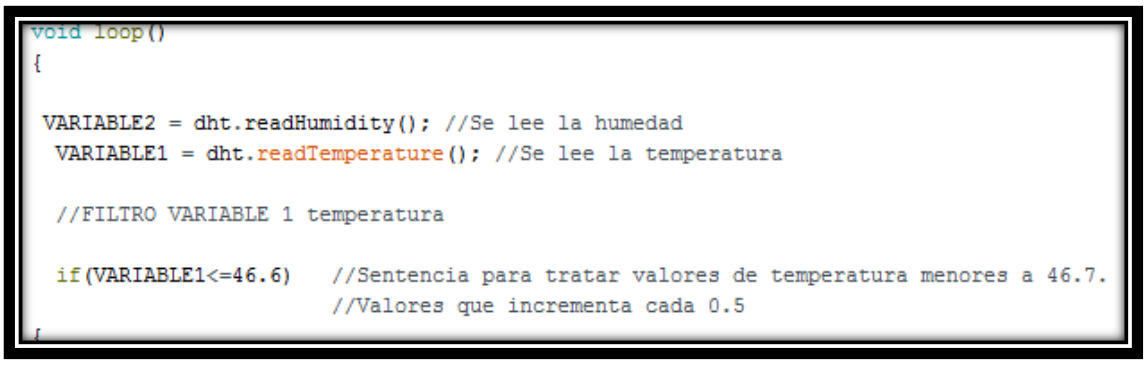


```
//Las siguientes variables serán usada para determinar el valor final que se le dará a cada variable.
//Teniendo en cuenta que en temperatura tendremos valores permitidos de -4 a +50 y
//en humedad de -4 a 130.
//En temperatura los valores se consideran cada 0.5, es decir, 0.5-1.0-1.5-2.0-2.5-3.0...
//Para los valores de humedad es la misma condición, pero apartir del valor 50,
//los valores incrementan de 5 en 5, hata el 130, es decir, 50-55-60-65...

int ENTERO1;//Variable para temperatura
int ENTERO2;//Variable para humedad
int RANGO1; //Variable para temperatura
int RANGO2; //Variable para temperatura
int APROXIMACION1; //Variable para temperatura
int APROXIMACION2; //Variable para temperatura
```

Figura 4.2 Programa arduino RX

Aquí se observa la función que se ocupa para capturar los valores del sensor y guardarlos en cada variable.



```
void loop()
{

 VARIABLE2 = dht.readHumidity(); //Se lee la humedad
  VARIABLE1 = dht.readTemperature(); //Se lee la temperatura

  //FILTRO VARIABLE 1 temperatura

  if(VARIABLE1<=46.6)    //Sentencia para tratar valores de temperatura menores a 46.7.
                         //Valores que incrementa cada 0.5
{
```

Figura 4.3 Programa arduino RX

Aquí se empieza el tratamiento del dato capturado, haremos uso de la sentencia “if” para tratar los distintos rangos de valores de los datos capturados por el sensor. Al realizar estos cálculos se usó la aproximación para valor obtenido, es decir, si el valor como se muestra en la imagen esta entre -4.9 y -4.5 se considerara que el valor es -4.5. Y así sucesivamente como se muestran en las dos primeras imágenes, se ha realizado una tarea extensa con los “if”, para poder dar un formato a nuestro valor, legible y con un solo número, evitando el error al momento de ser recibido por el receptor. Se recalca nuevamente que este problema es solo solucionable de esta forma, ya que no hay una matemática más simple para poder realizar estos tratamientos a los datos obtenidos, debido por la trasferencia de datos entre el arduino y la radio. Si hubiese ocupado un PLC, no tendríamos este problema que tenemos presente en este proyecto, pero por ensayos y casos prácticos en temas de accesibilidad, hemos ocupado el arduino para esta tesis.

```
if(VARIABLE1<1) // Sentencia que nos permite tratar lo valroes
                //de temperatura inferiores a 1
{
if((VARIABLE1>=-4.9)&&(VARIABLE1<=-4.5)) // Si el valor de temperatura está entre
                                          //los valores indicados, tomara el valor indicado en DATO1
DATO1=112;
if((VARIABLE1>=-4.4)&&(VARIABLE1<=-4))  // Si el valor de temperatura está entre
                                        //los valores indicados, tomara el valor indicado en DATO1
DATO1=59;
if((VARIABLE1>=-3.9)&&(VARIABLE1<=-3.5))  // Si el  de temperatura valor está entre
                                           //los valores indicados, tomara el valor indicado en DATO1
DATO1=111;
if((VARIABLE1>=-3.4)&&(VARIABLE1<=-3))  // Si el valor de temperatura está entre
                                        //los valores indicados, tomara el valor indicado en DATO1
DATO1=58;
```

Figura 4.4 Programa arduino RX

```
if (VARIABLE1>=1) //Sentencia que nos permite tratar los valroes de temperatura superiores a 0
{
ENTERO1= int (VARIABLE1);
RANGO1=(VARIABLE1-ENTERO1)*10;

if ((RANGO1>=0)&& (RANGO1<=3)) // Si el valor de temperatura está entre el rango indicado,
                               //tomará el valor indicado en DATO1
DATO1=(VARIABLE1-(RANGO1/10));
```

Figura 4.5 Programa arduino RX

…… (ver completo en anexo)

Finalmente en este recuadro se hará él envió de los datos, para ello se ha dividido en secciones él envió. Como vemos primero se envía el dato que indicara si la variable siguiente a enviar es temperatura o humedad (en primera instancia es temperatura), luego de un tiempo, se envía el dato de temperatura y de la misma forma se realiza con la humedad.

Cada “delay” ayuda a dar un tiempo al receptor para poder recibir los datos y luego el siguiente, de cierta formas se está sincronizando ambos arduinos.

```
 Serial.flush();                //Limpiamos el buffer de comunicación
Serial.write (temperatura);     // Enviamos por el puerto de comunicación RS232 el valor
                                //que indicará al receptor que se prepare para recibir datos de temperatura

 delay (1000);                  // Tiempo que damos para asegurarnos que el dato fue enviado y recibido
 Serial.write(DATO1);           //Enviamos el dato de valor de temperatura medida

 delay(2000);                   // Tiempo que damos para asegurarnos que el dato fue enviado y
                                //recibido antes de seguir a enviar el dato de la otra variable

 Serial.flush();               //Limpiamos el buffer de comunicación

 Serial.write (humedad);       // Enviamos por el puerto de comunicación RS232 el valor que indicará
                               //al receptor que se prepare para recibir dato de temperatura

 delay(1000);                  // Tiempo que damos para asegurarnos que el dato fue enviado y recibido
Serial.write(DATO2);           //Enviamos el dato de valor de temperatura medida

 delay (2000);                 // Tiempo que damos para asegurarnos que el dato fue enviado y
                               //recibido antes de seguir a enviar el dato de la otra variable


}
```

Figura 4.6 Programa arduino RX

4.2 Desarrollo de programación de arduino Mega 256 como equipo de recepción de variable y de mostrar valores en pantalla

Se empieza la programación agregando la librería para la pantalla LCD, y a su vez se define los pines que se ocuparan para conectar la pantalla LCD al arduino. Seguidamente se puede ver la variable “Datorx…” la cual corresponde a valores de una tabla que se ha definido en la programación del receptor, con la cual se interpretan los valores que envié el transmisor.

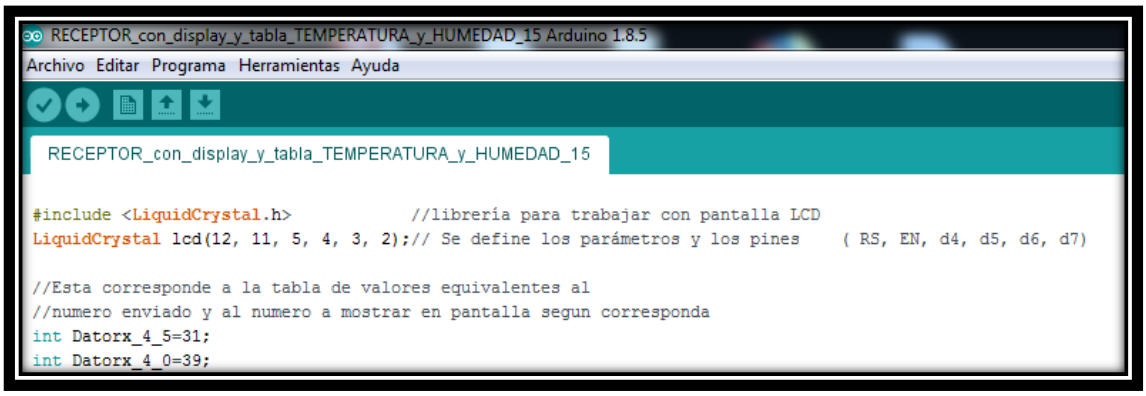


Figura 4.7 Programa arduino TX

Aquí puntualmente se presentan algunas variables definidas para poder guardar los datos recibido del transmisor y a su vez el dato que identificara a que medición corresponde, ya sea temperatura o humedad.

```
int VARIABLE=0;     // Variable que guardará el dato recibido
                    //de la medición de temperatura o humedad segun corresponda

int T=0;            // Variable que guaradará el valor que nos indicara
                    //si el dato enviado es de humedad o temperatura.

float humedad=0;    // Variable que guarda el valor de humedad
float equivalente=0; // Variable que guarda el valor de temperatura
```

Figura 4.8 Programa arduino TX

Una vez que se reciben los dos datos que espera el receptor, se empieza decodificar los datos, considerando el dato que define la medición que se recibe y la segunda el

valor de esa medición. Y una vez recibido el segundo dato se empieza el proceso de decodificación, como se puede ver claramente en la imagen. Para que se entiendan mejor estos valores será necesario que vayamos a la imagen que está en anexo Tabla valores.

```
{T=Serial1.read();              //Guarda el valor del primer dato que nos indicará a
                                //que variable corresponde el segundo dato recibido ( temperatura o humedad)
VARIABLE=Serial1.read();        //Guarda el valor del segundo dato recibido que corresponde a una variable sensada

// Estas sentencia no son mas que nada el equivalente del dato de variable enviada
//y la que se mostrara finalmente en pantalla, cada valor tiene su equivalente. Ver tabla.
if (VARIABLE==Datorx_4_5)
{equivalente=-4.5;}
if (VARIABLE==Datorx_4_0)
{equivalente=-4;}
if (VARIABLE==Datorx_3_5)
{equivalente=-3.5;}
```

…… (ver completo en anexo)

Figura 4.9 Programa arduino TX

Esta imagen se ha dejado para mostrar algo puntual, la variable humedad toma el mismo valor que tiene la variable “equivalente”, esto debido a que la variable equivalente tiene el valor final de la decodificación y esta variable se usa para mostrar el dato siempre y cuando el dato corresponda a temperatura, pero si es el caso contrario, entonces la variable humedad mostrara el valor de humedad, por ende se ha establecido que tengan el mismo valor final decodificado, pero la filtramos con el primer valor recibido que menciona a que medición corresponde dicho dato decodificado.

```
humedad=equivalente;}       // Aqui guardamos el segundo valor enviado,
                            //en la variable  temperatura y humedad, para
                            //luego definir a que variable corresponde el dato.

if(T==14)                   // Si el dato del primer valor enviado es 14 (ver tabala de valores equivalentes)
                            //el segundo dato sera de la variable temperatura, de lo contrario sera humedad.
{                           // Y procedemos a mostrarlo en pantalla
```

Figura 4.10 Programa arduino TX

En esta últimas imágenes se puede ver lo mencionado en la imagen anterior, y aquí más que nada se ve el proceso con el cual se muestra en la pantalla LCD, el dato de temperatura y el de humedad, pero se ve que en cada caso se ha establecido la ubicación del cursor “lcd.setCursor” para no sobrescribir algunas de las dos mediciones, y así poder mostrar ambas al mismo tiempo. Y como en el caso del transmisor hay unos delay, el cual como se mencionó anteriormente, es más que nada para establecer algún tipo de sincronización entre ambos arduino.

```
lcd. setCursor ( 0 , 0 ) ;  // Fijamos la fila  y columna desde donde se emepzara a
                            //escribir el dato de la variable temperatura
lcd.print ( equivalente);   // Mostramos en pantalla el valor final de la variable  sensada de temperatura
lcd.print (" C");           // Colocamos en pantalla el simbolo de temperatura en grados celsius
delay(1990);                // Damos un tiempo antes de empezar a recibir el  valor de la variable humedad
}
else {              //Sentencia en caso que la variable("T") que define si el dato corresponde a humedad o
                    //temperatura no fuera 14, aqui empezariamos a mostrar el valor la variable humedad
```

Figura 4.11 Programa arduino TX

```
  lcd. setCursor ( 0 , 1 ) ; // Fijamos el cursor de la pantalla LCD en la columna 0 y fila 1, esta vez
                             //nos corrimos para no borrar el dato de temperatura mostrada en pantalla,
                             //para asi mostrar los dos valores
lcd.print ( humedad);       // Mostramos en pantalla el valor final de la variable  sensada de humedad
lcd.print (" %");          // Colocamos en pantalla el simbolo de humedad en porcentaje
delay (1990);              // Damos un tiempo antes de empezar a recibir el  valor de la variable temperatura otra vez
}
}
```

Figura 4.12 Programa arduino TX

Capítulo 5

5. Configuración de equipos de radio.

Aquí se ve la primera configuración de la radio, la cual es de suma importancia, debido que se define el ID del radio, que vendría siendo como el nombre del equipo, el cual se empleara para poder saber a quién se envía el dato (en el caso del transmisor) y a su vez también indica de quien estamos recibiendo el dato (en el caso del receptor). Esta ID es posible asignar aleatoriamente según sea necesario, en este caso se ha ocupado el ID 2066 y 2067:

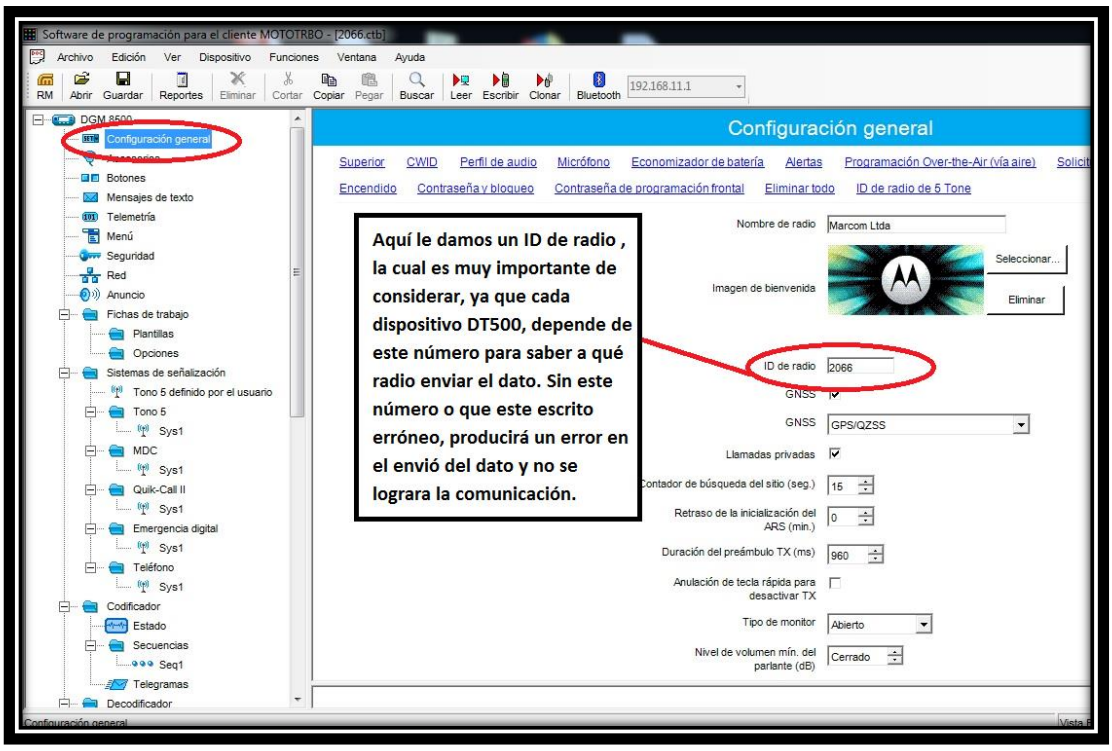


Figura 5.1 Configuración radio

Ahora en esta sección se le da privacidad al canal de comunicación, para ello se colocara un identificador al canal, el cual permitirá identificar si el dato enviado (esto es para el receptor) tiene el identificador del canal correcto para que el radio de el paso a la comunicación y capture el dato. Para el caso del transmisor, permite que solo aquella radio que tenga el mismo identificador de radio, pueda recibir su dato enviado. Con esto se puede destacar dos cosas, primeramente que ambas radios deben tener el mismo identificador del canal y segundo que con esto se puede evitar que interferencias o alguna señal cercana a los valores del canal entorpezcan la comunicación:

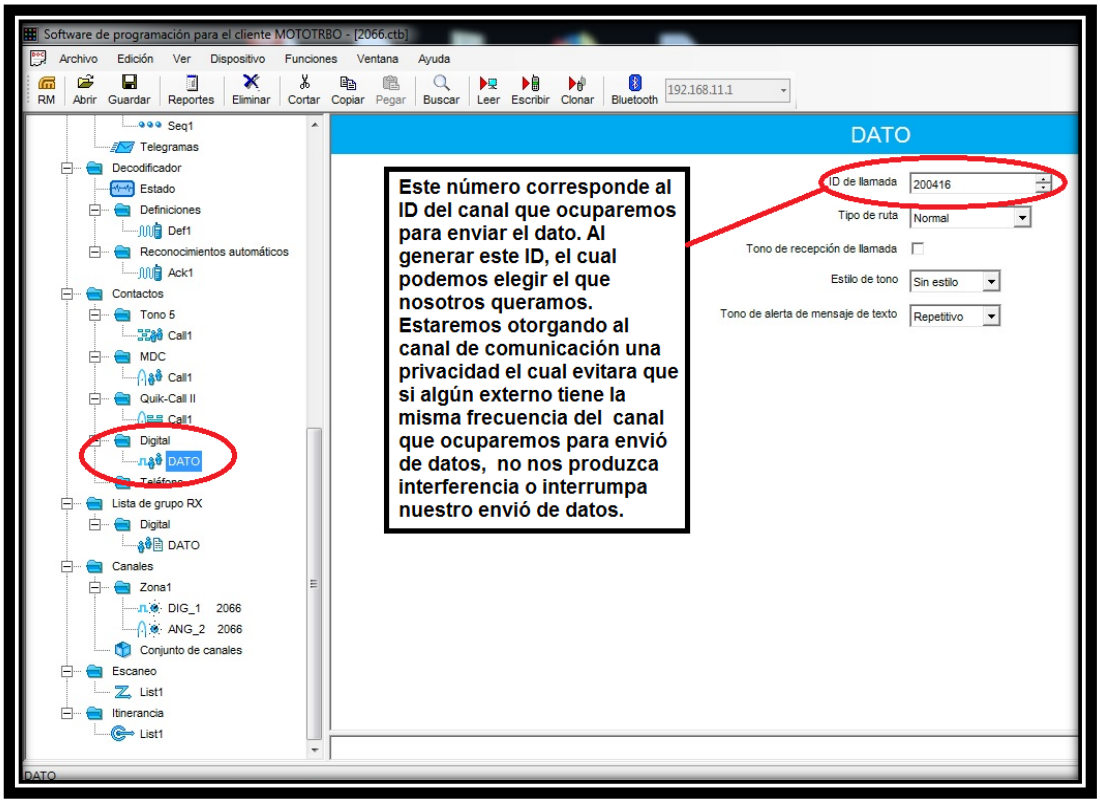


Figura 5.2 Configuración radio

Esta es otra configuración para la privacidad del canal de comunicación, aquí se asocia el identificador del canal a la parte de RX de la radio, es muy

necesario que este agregada el ID del canal a nuestra lista de RX, debido a que más adelante se ocupara para indicar que para la recepción de la radio, esta solo recibirá datos que contengas este ID de canal y para el transmisor solo enviara el dato aquellas radio que contengan este identificador de canal :

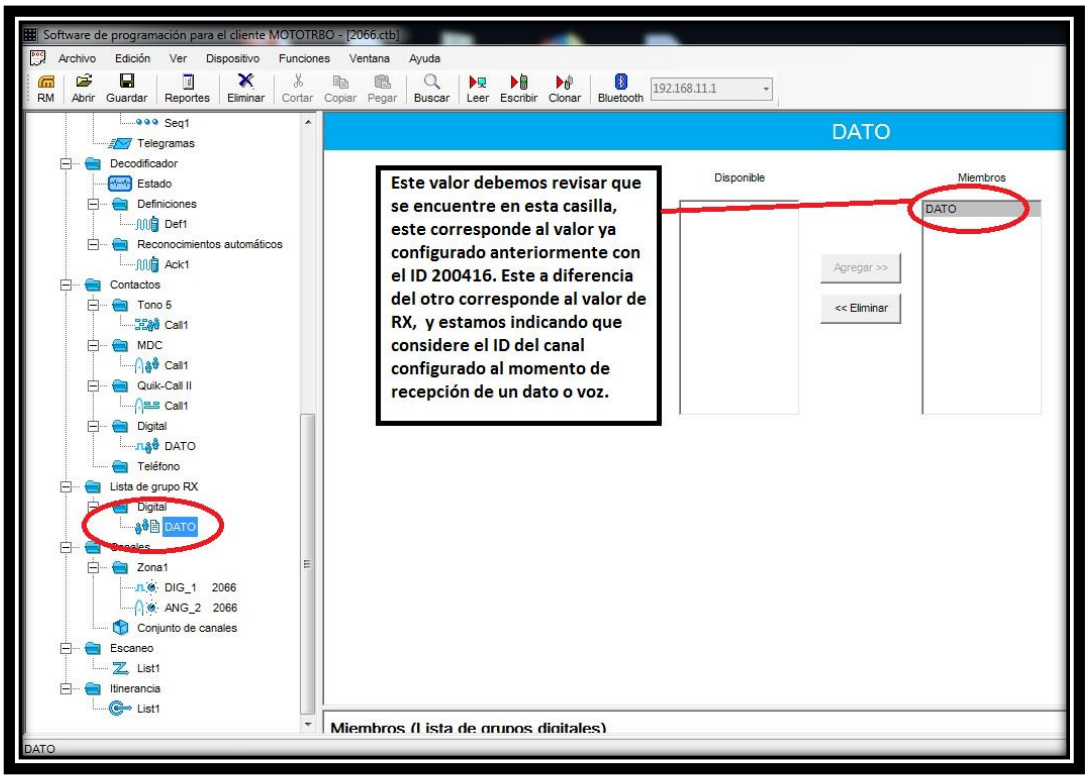


Figura 5.3 Configuración radio

Aquí se configura el parámetro del canal digital que se ocupara para transmitir los datos. Cabe señalar que estos valor también proporcionan privacidad al canal, pero de una forma menos segura que a la que ya se ha mencionado,

porque es más fácil que algún equipo tenga justo el mismo código de color y franja de tiempo repetidor, pero más difícil es que coincida que el ID de canal sea el mismo, ya que es un valor que se ha inventado, y estos que se tienen aquí son valores predefinido por la radio, que varían e 1 a 15 (código de color) y 1 y 2 (franja de tiempo repetidor). En este caso por elección se ha definido el código de color 4 y franja de tiempo repetidor 2:

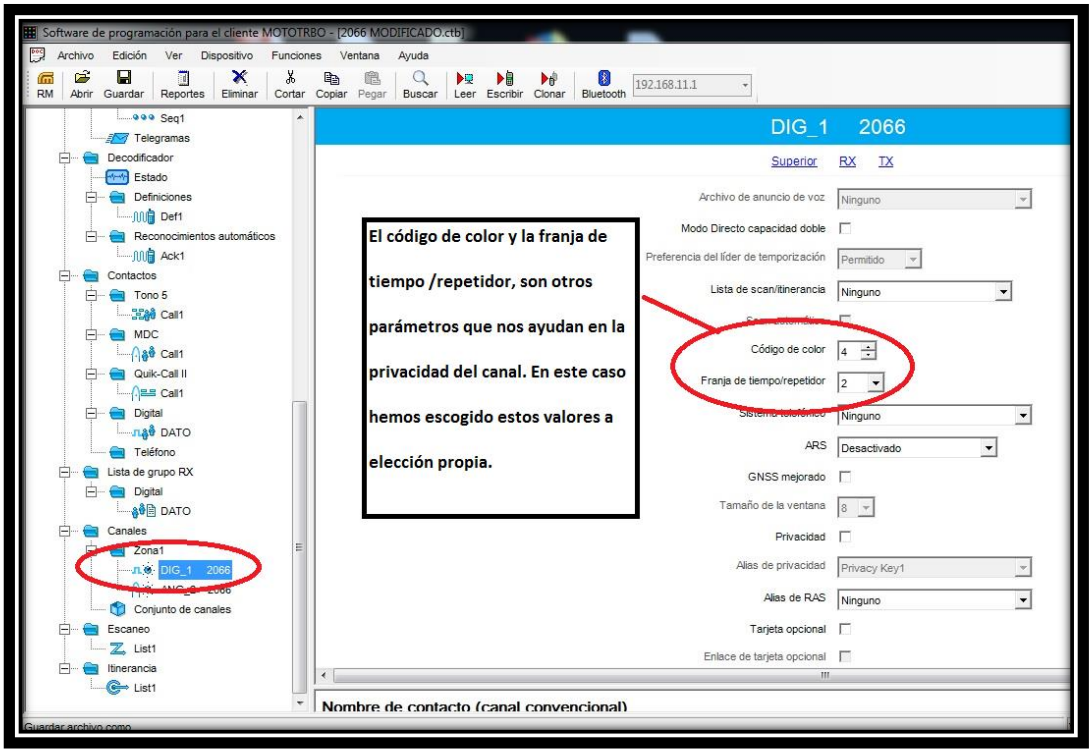


Figura 5.4

En esta ventana se escriben las frecuencias que se ocupan, en este caso de un rango de UHF, porque UHF esto es debido que esta rango del espectro de frecuencia se comporta de mejor forma al momento de recorrer distancias largas con obstáculos de por medio. Al ser UHF también se disminuye la potencia del equipo, alcanzando distancia iguales que las de frecuencias de

rango VHF, pero esta última usa mayor potencia al transmitir, aumentando costos en sus equipos. La frecuencia UHF para este equipo varía entre 450.000 – 527.000. Para este proyecto se ha definido 465.025 para RX y TX:

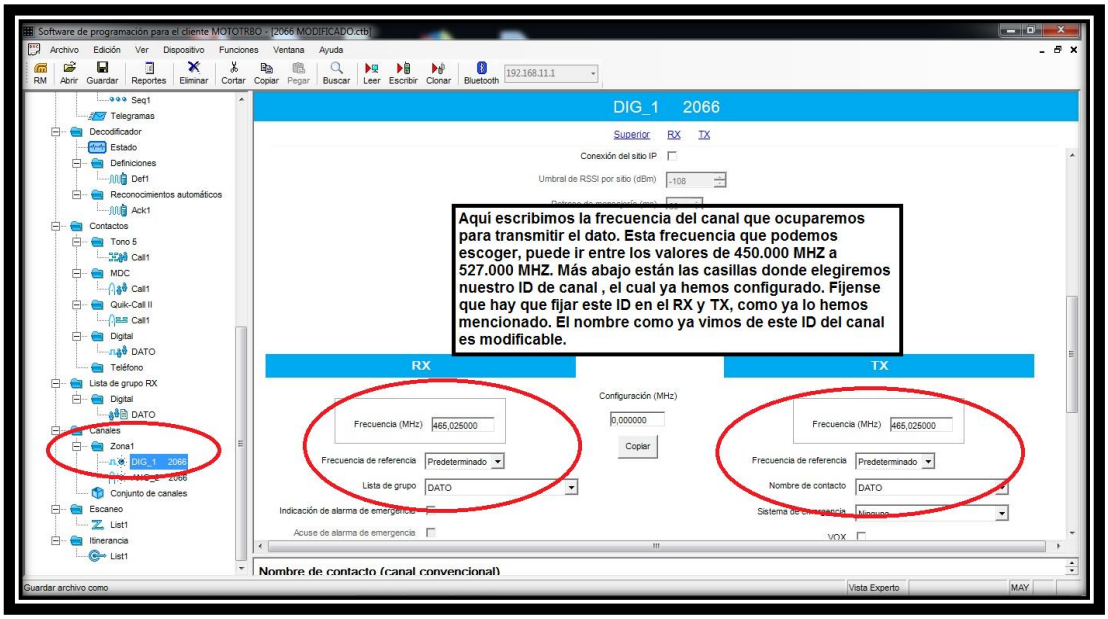


Figura 5.5 Configuración radio

Capítulo 6

6.Configuracion del DT500

Se comienza configurando la velocidad de comunicación. Se ha fijado esta velocidad porque es la que se adecua para nosotros, dado a que no se necesita mayor velocidad, porque las variables no presentan variaciones tan rápidas como para requerir mayor rapidez en la comunicación :

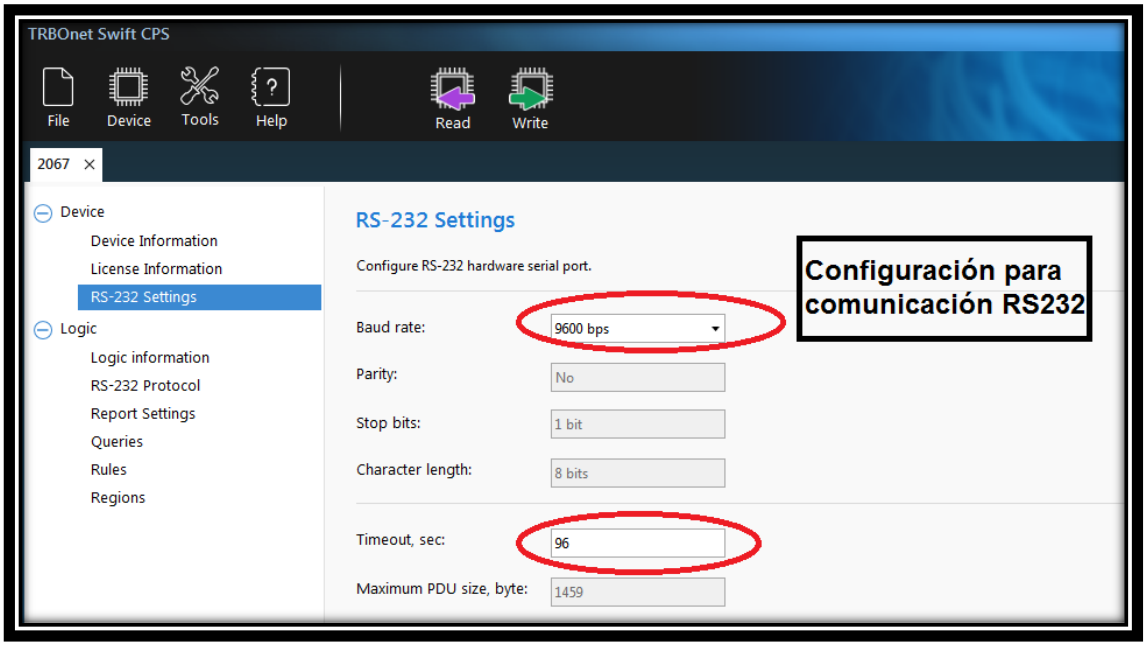


Figura 6.1 Configuración DT500

Se selecciona el formato en que se tratara el dato. Se ha elegido RAW DATA, dado que en esta configuración se ha logrado establecer la comunicación, debido que las otras opciones no aplican para el uso que se le dará al DT500 y con esta opción se hace más fácil enlazar el arduino con la radio:



Figura 6.2 Configuración DT500

Se Accede a la configuración del puerto serial:

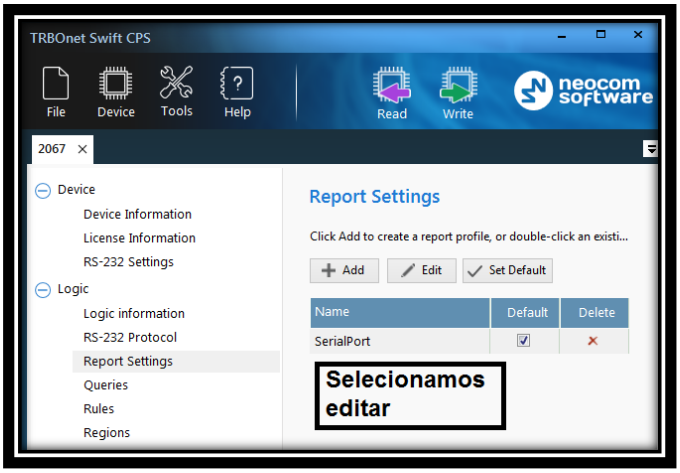


Figura 6.3 Configuración DT500

Finalmente se habilita y configura el valor que enlazaran las radios con los dispositivos DT500. Se puede ver que aquí se empieza asignar los valores de la radio al DT500, tanto el valor de red e ID de radio. El valor de la red viene por defecto en la radio en sus configuraciones. En donde si hay que tener cuidado, es al colocar el ID, puesto que aquí se coloca el ID al cual la radio se comunicara, es decir, si el DT500 está conectado a la radio con ID 2067, por ende se enviara el dato a la radio con ID 2066 (según como se configuren las radios), entonces se coloca dicho ID como se ve en la imagen, aquí se está indicando que esta radio con ID 2067 enviara el dato a la radio 2066:

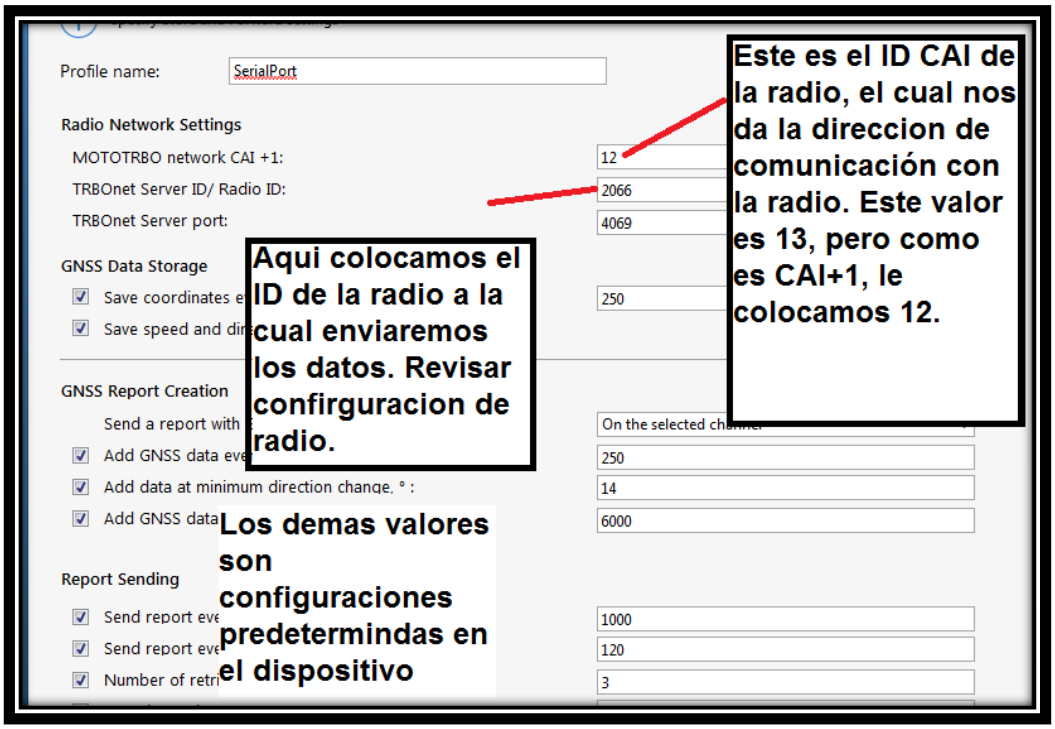


Figura 6.4 Configuración DT500

Capítulo 7

7. Puesta en marcha

Conexiones sensor al arduino uno :

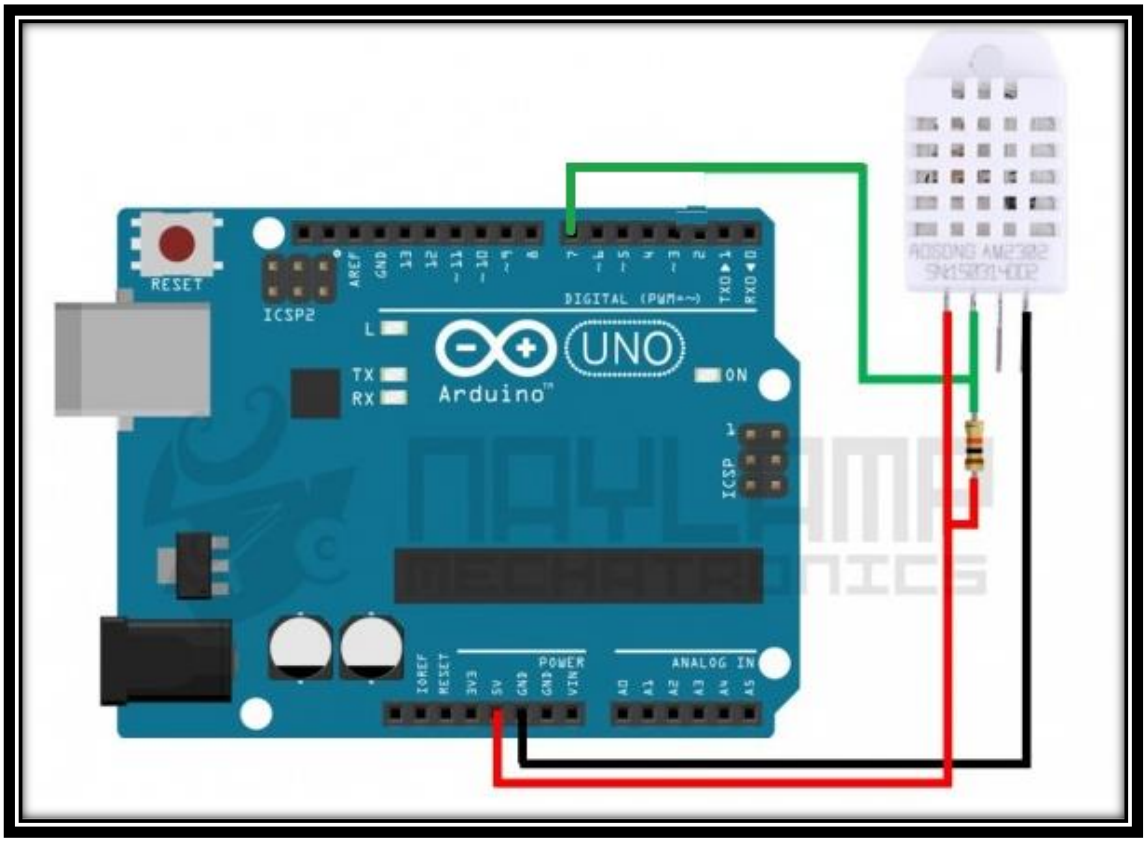


Figura 7.1 Conexión sensor y arduino uno

Conexión LCD a arduino mega:

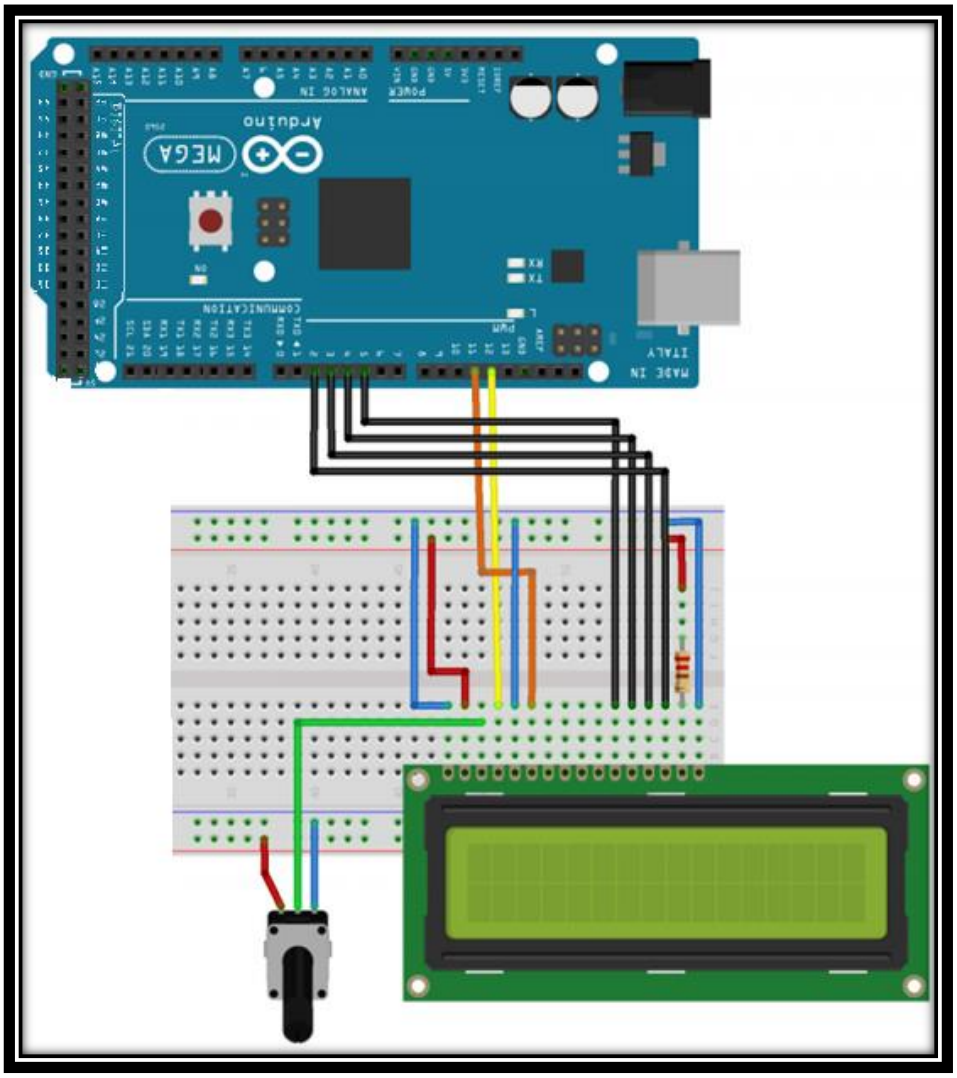


Figura 7.2 Conexión arduino mega y pantalla LCD

A continuación se presentan las conexiones del transmisor, contemplando las conexiones de sensor al arduino, arduino a DT500 y Dt 500 a Radio Motorola. También se ven las demás conexiones de fuente y antena.

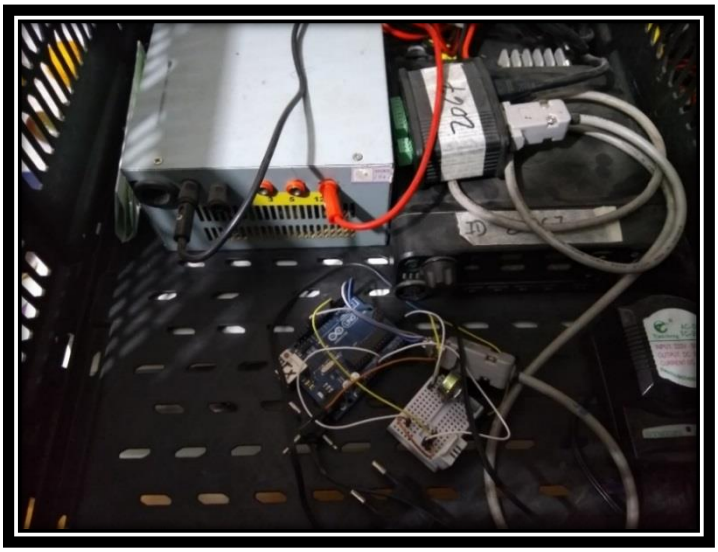


Figura 7.3 Equipo transmisor

Ahora aquí se presentan las conexiones del receptor, contemplando las conexiones del LCD al arduino, arduino a DT500 y Dt 500 a Radio Motorola. También se ven las demás conexiones de fuente y antena.

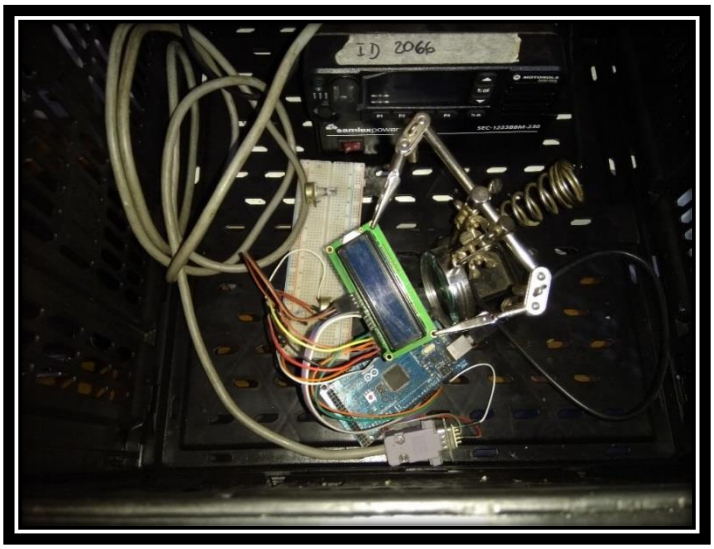


Figura 7.4 Equipo receptor



Figura 7.5 Equipo receptor


SENSOR DHT-22
RADIO DGM 8500 UHF (TRANSMISOR)
ANTENA
ARDUINO UNO R3
CONECTOR DT500 A PUERTO TRASERO DE RADIO DGM 8500
PROPAGACION POR AIRE
CABLE RS232
DT 500 TRBONET
PANTALLA LCD 16X2
ARDUINO MEGA 2560 R3
ANTENA
CABLE RS232
RADIO DGM 8500 UHF (RECEPTOR)
DT 500 TRBONET
CONECTOR DT500 A PUERTO TRASERO DE RADIO DGM 8500

Conclusión

Resultados obtenidos:

- Se ha expuesto que si existe otra alternativa a considerar a la hora de pensar en: ¿cómo se podrá monitorear una variable remotamente?, ¿cómo se hará él envió de datos en largas distancia? Y ¿cómo se puede abaratar costos? , preguntas que siempre surgen, más aun cuando el lugar geográfico es muy complejo, ya sea por distancias largas, temperaturas extremas, terrenos de difícil acceso, etc.
- Se ha podido aprender cómo estos equipos de radio cumplen características de altos estándares como lo es el estándar militar en trabajos en ambientes hostiles y de temperaturas extremas.
- Se conoció el dispositivo DT500 que es la interfaz entre la radio y el arduino en este caso (podría ser también otro equipo como el PLC), con el cual se establece la unión de dos equipos muy distintos, pero que ayudan a dar solución a un tema muy importante que es poder monitorear remotamente variables o datos a largas distancias y agilizando aun mayor el trabajo de áreas de proceso o control de residuos o combustible.
- Finalmente se pudo demostrar que si es posible usar la señal RF para enviar datos a larga distancia, sin uso de cables.

Bibliografía

- ✓ Accesorios equipos de radio
- ✓ Conectores para equipos RF
- ✓ **Datasheet DT500**
- ✓ **Datasheet equipo de radio**
- ✓ **Conexión del sensor al arduino**
- ✓ **Conexión pantalla LCD al Arduino**

Anexos

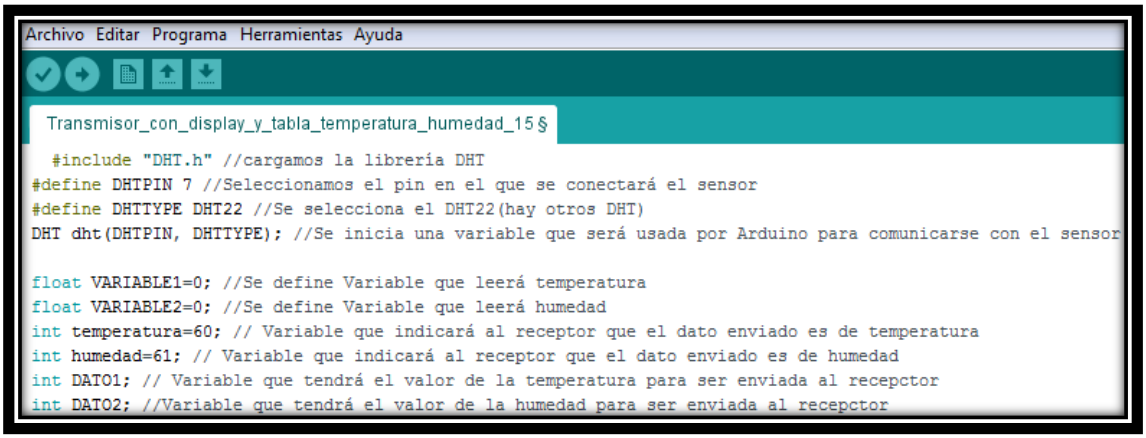
Tabla de valores para arduino receptor:

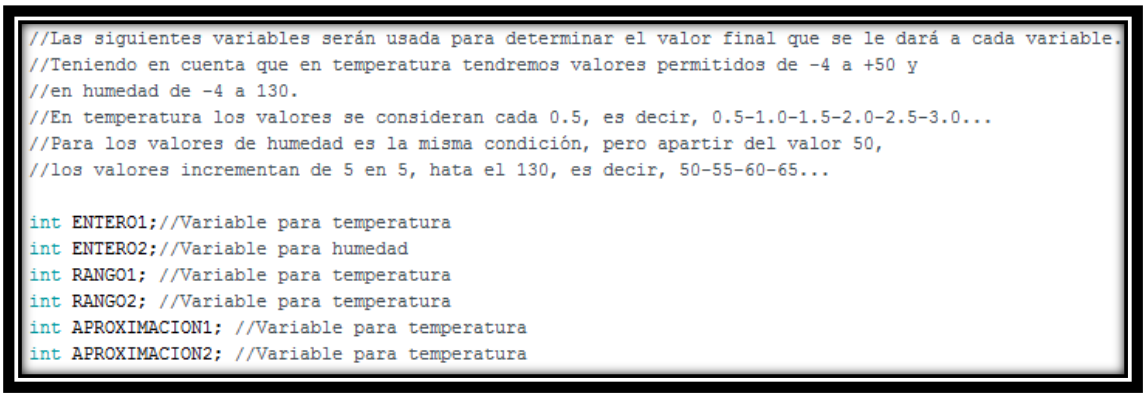
NUMERO ENVIADO	NUMERO RECIBIDO EN ARDUINO	EQUIVALENTE FINAL A MOSTRAR EN PANTALLA
112	31	-4,5
59	39	-4
111	11	-3,5
58	103	-3
110	27	-2,5
57	78	-2
109	91	-1,5
56	30	-1
55	19	0
108	59	0.5
1	64	1
63	2	1.5
2	96	2
64	127	2.5
3	32	3
65	80	3.5
4	112	4
66	104	4.5
5	65	5
67	40	5.5
6	48	6
68	116	6.5
7	16	7
69	81	7.5
8	120	8
70	52	8.5
9	66	9
71	20	9.5
10	97	10
72	122	10.5
11	33	11
73	82	11.5
12	56	12
74	105	12.5
13	67	13
75	41	13.5
14	24	14
76	58	14.5
15	8	15

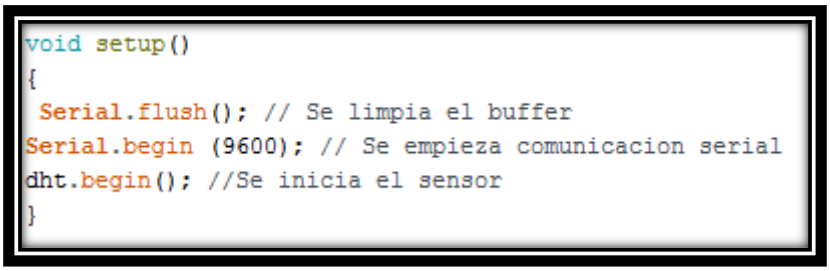
77	83	15.5
16	124	16
78	26	16.5
17	68	17
79	10	17.5
18	98	18
80	125	18.5
19	34	19
81	84	19.5
20	113	20
82	106	20.5
21	69	21
83	42	21.5
22	49	22
84	117	22.5
23	17	23
85	85	23.5
24	60	24
86	53	24.5
25	70	25
87	21	25.5
26	99	26
88	61	26.5
27	35	27
89	86	27.5
28	28	28
90	107	28.5
29	71	29
91	43	29.5
30	12	30
92	29	30.5
31	4	31
93	87	31.5
32	126	32
94	13	32.5
33	72	33
95	5	33.5
34	100	34
96	63	34.5
35	36	35
97	88	35.5
36	114	36
98	108	36.5
37	73	37
99	44	37.5

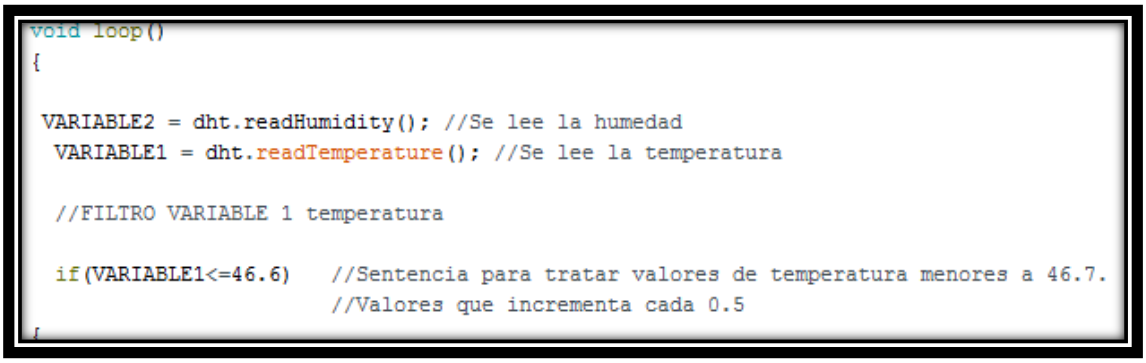
38	50	38
100	118	38.5
39	18	39
101	89	39.5
40	121	40
102	54	40.5
41	74	41
103	22	41.5
42	101	42
104	123	42.5
43	37	43
105	90	43.5
44	57	44
106	109	44.5
45	75	45
107	45	45.5
46	25	46
47	9	46.5
48	62	47
49	76	47.5
50	102	48
51	38	48.5
52	115	49
53	77	49.5
54	51	50
61	79	55
62	6	60
113	92	65
114	110	70
115	46	75
116	119	80
117	93	85
118	55	90
119	23	95
120	15	100
121	94	105
122	111	110
123	47	115
124	7	120
125	95	125
126	3	130

Programación Arduino transmisor:









```
if(VARIABLE1<1) // Sentencia que nos permite tratar lo valroes
                //de temperatura inferiores a 1
{
if((VARIABLE1>=-4.9)&&(VARIABLE1<=-4.5)) // Si el valor de temperatura está entre
                                          //los valores indicados, tomara el valor indicado en DATO1
DATO1=112;
if((VARIABLE1>=-4.4)&&(VARIABLE1<=-4))  // Si el valor de temperatura está entre
                                         //los valores indicados, tomara el valor indicado en DATO1
DATO1=59;
if((VARIABLE1>=-3.9)&&(VARIABLE1<=-3.5))  // Si el  de temperatura valor está entre
                                           //los valores indicados, tomara el valor indicado en DATO1
DATO1=111;
if((VARIABLE1>=-3.4)&&(VARIABLE1<=-3))  // Si el valor de temperatura está entre
                                         //los valores indicados, tomara el valor indicado en DATO1
DATO1=58;
```

```
if((VARIABLE1>=-2.9)&&(VARIABLE1<=-2.5)) // Si el valor de temperatura está entre
                                          //los valores indicados, tomara el valor indicado en DATO1
DATO1=110;
if((VARIABLE1>=-2.4)&&(VARIABLE1<=-2))  // Si el valor de temperatura está entre
                                         //los valores indicados, tomara el valor indicado en DATO1
DATO1=57;
if((VARIABLE1>=-1.9)&&(VARIABLE1<=-1.5)) // Si el valor de temperatura está entre
                                          //los valores indicados, tomara el valor indicado en DATO1
DATO1=109;
if((VARIABLE1>=-1.4)&&(VARIABLE1<=-1))  // Si el valor de temperatura está entre
                                         //los valores indicados, tomara el valor indicado en DATO1
DATO1=56;
if((VARIABLE1>=0.5)&&(VARIABLE1<=0.9)) // Si el valor de temperatura está entre
                                       //los valores indicados, tomara el valor indicado en DATO1
DATO1=108;
if((VARIABLE1>=0)&&(VARIABLE1<=0.4)) // Si el valor de temperatura está entre
                                     //los valores indicados, tomara el valor indicado en DATO1
DATO1=55;
}
```

```
if (VARIABLE1>=1) //Sentencia que nos permite tratar los valroes de temperatura superiores a 0
{
ENTERO1= int (VARIABLE1);
RANGO1=(VARIABLE1-ENTERO1)*10;

if ((RANGO1>=0)&& (RANGO1<=3)) // Si el valor de temperatura está entre el rango indicado,
                               //tomará el valor indicado en DATO1
DATO1=(VARIABLE1-(RANGO1/10));
```

```
if ((RANGO1>=7)&&(RANGO1<=9)) // Si el valor de temperatura  está entre el rango indicado,
                               //tomará el valor indicado en DATO1
DATO1=(VARIABLE1-(RANGO1/10))+1;

if((RANGO1>=4)&&(RANGO1<=6)) // Sentencia que trabajara los valores de temperatura
                             //que terminen en .5 ejemplo 4.5
{
  APROXIMACION1= ENTERO1; // Aqui se quita el decimal y se deja el entero, esto es para
                          //facilitar la operación de sentencias. APROXIMACION1==1,
                          //quiere decier 1.5. APROXIMACION1==2, quiere decier 2.5. etc..
```

```
if (APROXIMACION1==1)  // Si el valor de temperatura es 1.5, tomará el valor de DATO1
DATO1=63;
if (APROXIMACION1==2)  // Si el valor de temperatura es 2.5, tomará el valor de DATO1
DATO1=64;
if (APROXIMACION1==3) // Si el valor de temperatura es 3.5, tomará el valor de DATO1
DATO1=65;
if (APROXIMACION1==4) // Siel valor de temperatura es 4.5, tomará el valor de DATO1
DATO1=66;
if (APROXIMACION1==5) // Si el valor de temperatura es 5.5, tomará el valor de DATO1
DATO1=67;
if (APROXIMACION1==6) // Si el valor de temperatura es 6.5, tomará el valor de DATO1
DATO1=68;
if (APROXIMACION1==7) // Si el valor de temperatura es 7.5, tomará el valor de DATO1
DATO1=69;
if (APROXIMACION1==8) //Si el valor de temperatura es 8.5, tomará el valor de DATO1
DATO1=70;
```

```
if (APROXIMACION1==9) //Si el valor de temperatura es 9.5, tomará el valor de DATO1
DATO1=71;
if (APROXIMACION1==10) // Si el valor de temperatura es 10.5, tomará el valor de DATO1
DATO1=72;
if (APROXIMACION1==11) //Si el valor de temperatura es 11.5, tomará el valor de DATO1 etc.....
DATO1=73;
if (APROXIMACION1==12)
```

```
DATO1=74;
if (APROXIMACION1==13)
DATO1=75;
if (APROXIMACION1==14)
DATO1=76;
if (APROXIMACION1==15)
DATO1=77;
if (APROXIMACION1==16)
DATO1=78;
if (APROXIMACION1==17)
DATO1=79;
if (APROXIMACION1==18)
DATO1=80;
if (APROXIMACION1==19)
DATO1=81;
if (APROXIMACION1==20)
DATO1=82;
if (APROXIMACION1==21)
DATO1=83;
if (APROXIMACION1==22)
DATO1=84;
```

```
if (APROXIMACION1==23)
DATO1=85;
if (APROXIMACION1==24)
DATO1=86;
if (APROXIMACION1==25)
DATO1=87;
if (APROXIMACION1==26)
DATO1=88;
if (APROXIMACION1==27)
DATO1=89;
if (APROXIMACION1==28)
DATO1=90;
if (APROXIMACION1==29)
DATO1=91;
if (APROXIMACION1==30)
DATO1=92;
if (APROXIMACION1==31)
DATO1=93;
if (APROXIMACION1==32)
DATO1=94;
if (APROXIMACION1==33)
DATO1=95;
if (APROXIMACION1==34)
DATO1=96;
if (APROXIMACION1==35)
DATO1=97;
if (APROXIMACION1==36)
DATO1=98;
```

```
if (APROXIMACION1==37)
DATO1=99;
if (APROXIMACION1==38)
DATO1=100;
if (APROXIMACION1==39)
DATO1=101;
if (APROXIMACION1==40)
DATO1=102;
if (APROXIMACION1==41)
DATO1=103;
if (APROXIMACION1==42)
DATO1=104;
if (APROXIMACION1==43)
DATO1=105;
if (APROXIMACION1==44)
DATO1=106;
if (APROXIMACION1==45)
DATO1=107;
if (APROXIMACION1==46)
DATO1=47;
if (APROXIMACION1==47)
DATO1=49;
if (APROXIMACION1==48)
DATO1=51;
if (APROXIMACION1==49)
DATO1=53;
if (APROXIMACION1==50)
DATO1=54;
```

```
}
}
}

//Aqui se trataran los valroes sobre 50, los cuales avnazan de 5 en 5


if((VARIABLE1>=51)&&(VARIABLE1<=55))  // Si el valor de temperatura está entre
                                       //los valores indicados, tomara el valor indicado en DATO1
DATO1=61;
if((VARIABLE1>=56)&&(VARIABLE1<=60)) // Si el valor de temperatura está entre
                                     //los valores indicados, tomara el valor indicado en DATO1
DATO1=62;
if((VARIABLE1>=61)&&(VARIABLE1<=65)) // Si el valor de temperatura está entre
                                     //los valores indicados, tomara el valor indicado en DATO1
DATO1=113;
if((VARIABLE1>=66)&&(VARIABLE1<=70)) // Si el valor de temperatura está entre
                                     //los valores indicados, tomara el valor indicado en DATO1
DATO1=114;
```

```
if((VARIABLE1>=71)&&(VARIABLE1<=75)) // Si el valor de temperatura está entre
                                     //los valores indicados, tomara el valor indicado en DATO1
DATO1=115;
if((VARIABLE1>=76)&&(VARIABLE1<=80)) // Si el valor de temperatura está entre
                                     //los valores indicados, tomara el valor indicado en DATO1
DATO1=116;
if((VARIABLE1>=81)&&(VARIABLE1<=85)) // Si el valor de temperatura está
                                     //entre los valores indicados, tomara el valor indicado en DATO1
DATO1=117;
```

```
if((VARIABLE1>=86)&&(VARIABLE1<=90)) // Si el valor de temperatura está entre
                                      //los valores indicados, tomara el valor indicado en DATO1
DATO1=118;
if((VARIABLE1>=91)&&(VARIABLE1<=95)) // Si el valor de temperatura está entre
                                      //los valores indicados, tomara el valor indicado en DATO1
DATO1=119;
if((VARIABLE1>=96)&&(VARIABLE1<=100)) // Si el valor de temperatura está entre
                                       //los valores indicados, tomara el valor indicado en DATO1
DATO1=120;
if((VARIABLE1>=101)&&(VARIABLE1<=105)) // Si el valor de temperatura está entre
                                        //los valores indicados, tomara el valor indicado en DATO1
DATO1=121;
if((VARIABLE1>=106)&&(VARIABLE1<=110)) // Si el valor de temperatura está entre
                                        //los valores indicados, tomara el valor indicado en DATO1
```

```
 DATO1=122;
 if((VARIABLE1>=111)&&(VARIABLE1<=115)) // Si el valor de temperatura está entre
                                        //los valores indicados, tomara el valor indicado en DATO1
 DATO1=123;
 if((VARIABLE1>=116)&&(VARIABLE1<=120)) // Si el valor de temperatura está entre
                                        //los valores indicados, tomara el valor indicado en DATO1
 DATO1=124;
 if((VARIABLE1>=121)&&(VARIABLE1<=125)) // Si el valor de temperatura está entre
                                        //los valores indicados, tomara el valor indicado en DATO1
 DATO1=125;
 if((VARIABLE1>=126)&&(VARIABLE1<=130)) // Si el valor de temperatura está entre
                                        //los valores indicados, tomara el valor indicado en DATO1
 DATO1=126;
```

```
//Para el FILTRO VARIABLE 2, se aplica lo mismo que el que ya mencionamos en temperatura,
//pero esta vez en ves de medir temperatura, estaremos midiendo humedad. Se uasa variable DATO2.

//FILTRO VARIABLE 2

 if(VARIABLE2<=46.6)
{

if(VARIABLE2<1)
{
if((VARIABLE2>=-4.9)&&(VARIABLE2<=-4.5))
DATO2=112;
if((VARIABLE2>=-4.4)&&(VARIABLE2<=-4))
DATO2=59;
if((VARIABLE2>=-3.9)&&(VARIABLE2<=-3.5))
DATO2=111;
if((VARIABLE2>=-3.4)&&(VARIABLE2<=-3))
DATO2=58;
```

```
if((VARIABLE2>=-2.9)&&(VARIABLE2<=-2.5))
DATO2=110;
if((VARIABLE2>=-2.4)&&(VARIABLE2<=-2))
DATO2=57;
if((VARIABLE2>=-1.9)&&(VARIABLE2<=-1.5))
DATO2=109;
if((VARIABLE2>=-1.4)&&(VARIABLE2<=-1))
DATO2=56;
if((VARIABLE2>=0.5)&&(VARIABLE2<=0.9))
DATO2=108;
```

```
if((VARIABLE2>=0)&&(VARIABLE2<=0.4))
DATO2=55;
}

if (VARIABLE2>=1)
{
ENTERO2= int (VARIABLE2);
RANGO2=(VARIABLE2-ENTERO2)*10;

if ((RANGO2>=0)&& (RANGO2<=3))
DATO2=(VARIABLE2-(RANGO2/10));

if ((RANGO2>=7)&&(RANGO2<=9))
DATO2=(VARIABLE2-(RANGO2/10))+1;

if((RANGO2>=4)&&(RANGO2<=6))
{
    APROXIMACION2= ENTERO2;


if (APROXIMACION2==1)
DATO2=63;
if (APROXIMACION2==2)
DATO2=64;
if (APROXIMACION2==3)
DATO2=65;
if (APROXIMACION2==4)
DATO2=66;
```

```
if (APROXIMACION2==5)
DATO2=67;
if (APROXIMACION2==6)
DATO2=68;
if (APROXIMACION2==7)
DATO2=69;
if (APROXIMACION2==8)
DATO2=70;
if (APROXIMACION2==9)
DATO2=71;
if (APROXIMACION2==10)
DATO2=72;
if (APROXIMACION2==11)
DATO2=73;
if (APROXIMACION2==12)
DATO2=74;
if (APROXIMACION2==13)
DATO2=75;
if (APROXIMACION2==14)
DATO2=76;
if (APROXIMACION2==15)
DATO2=77;
if (APROXIMACION2==16)
DATO2=78;
if (APROXIMACION2==17)
DATO2=79;
if (APROXIMACION2==18)
DATO2=80;
```

```
if (APROXIMACION2==19)
DATO2=81;
if (APROXIMACION2==20)
DATO2=82;
if (APROXIMACION2==21)
DATO2=83;
if (APROXIMACION2==22)
DATO2=84;
if (APROXIMACION2==23)
DATO2=85;
if (APROXIMACION2==24)
DATO2=86;
if (APROXIMACION2==25)
DATO2=87;
if (APROXIMACION2==26)
DATO2=88;
if (APROXIMACION2==27)
DATO2=89;
if (APROXIMACION2==28)
DATO2=90;
if (APROXIMACION2==29)
DATO2=91;
if (APROXIMACION2==30)
DATO2=92;
if (APROXIMACION2==31)
DATO2=93;
if (APROXIMACION2==32)
DATO2=94;
```

```
if (APROXIMACION2==33)
DATO2=95;
if (APROXIMACION2==34)
DATO2=96;
if (APROXIMACION2==35)
DATO2=97;
if (APROXIMACION2==36)
DATO2=98;
if (APROXIMACION2==37)
DATO2=99;
if (APROXIMACION2==38)
DATO2=100;
if (APROXIMACION2==39)
DATO2=101;
if (APROXIMACION2==40)
DATO2=102;
if (APROXIMACION2==41)
DATO2=103;
if (APROXIMACION2==42)
DATO2=104;
if (APROXIMACION2==43)
DATO2=105;
if (APROXIMACION2==44)
DATO2=106;
if (APROXIMACION2==45)
DATO2=107;
if (APROXIMACION2==46)
DATO2=47;
```

```
if (APROXIMACION2==47)
DATO2=49;
if (APROXIMACION2==48)
DATO2=51;
if (APROXIMACION2==49)
DATO2=53;
}
}
}


if((VARIABLE2>=51)&&(VARIABLE2<=55))
DATO2=61;
if((VARIABLE2>=56)&&(VARIABLE2<=60))
DATO2=62;
if((VARIABLE2>=61)&&(VARIABLE2<=65))
DATO2=113;
if((VARIABLE2>=66)&&(VARIABLE2<=70))
DATO2=114;
if((VARIABLE2>=71)&&(VARIABLE2<=75))
DATO2=115;
if((VARIABLE2>=76)&&(VARIABLE2<=80))
DATO2=116;
if((VARIABLE2>=81)&&(VARIABLE2<=85))
DATO2=117;
if((VARIABLE2>=86)&&(VARIABLE2<=90))
DATO2=118;
```

```
if((VARIABLE2>=91)&&(VARIABLE2<=95))
DATO2=119;
if((VARIABLE2>=96)&&(VARIABLE2<=100))
DATO2=120;
if((VARIABLE2>=101)&&(VARIABLE2<=105))
DATO2=121;
if((VARIABLE2>=106)&&(VARIABLE2<=110))
DATO2=122;
if((VARIABLE2>=111)&&(VARIABLE2<=115))
DATO2=123;
if((VARIABLE2>=116)&&(VARIABLE2<=120))
DATO2=124;
if((VARIABLE2>=121)&&(VARIABLE2<=125))
DATO2=125;
if((VARIABLE2>=126)&&(VARIABLE2<=130))
DATO2=126;
```

```
 Serial.flush();                //Limpiamos el buffer de comunicación
Serial.write (temperatura);     // Enviamos por el puerto de comunicación RS232 el valor
                                //que indicará al receptor que se prepare para recibir datos de temperatura

 delay (1000);                  // Tiempo que damos para asegurarnos que el dato fue enviado y recibido
 Serial.write(DATO1);           //Enviamos el dato de valor de temperatura medida

 delay(2000);                   // Tiempo que damos para asegurarnos que el dato fue enviado y
                                //recibido antes de seguir a enviar el dato de la otra variable

 Serial.flush();               //Limpiamos el buffer de comunicación

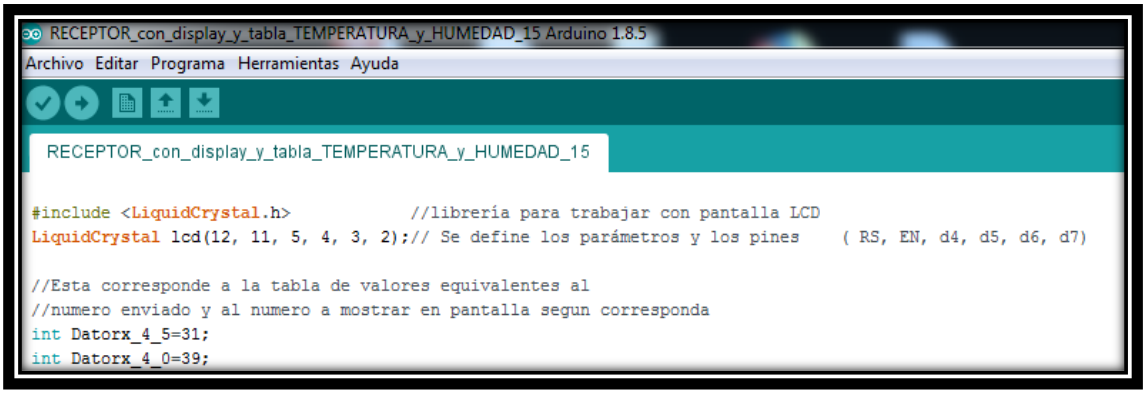
 Serial.write (humedad);       // Enviamos por el puerto de comunicación RS232 el valor que indicará
                               //al receptor que se prepare para recibir dato de temperatura

 delay(1000);                  // Tiempo que damos para asegurarnos que el dato fue enviado y recibido
Serial.write(DATO2);           //Enviamos el dato de valor de temperatura medida

 delay (2000);                 // Tiempo que damos para asegurarnos que el dato fue enviado y
                               //recibido antes de seguir a enviar el dato de la otra variable


}
```

Programación Arduino receptor:



```
int Datorx_3_5=11;
int Datorx_3_0=103;
int Datorx_2_5=27;
int Datorx_2_0=78;
int Datorx_1_5=91;
int Datorx_1_0=30;
int Datorx0=19;
int Datorx0_5=59;
int Datorx1=64;
int Datorx1_5=2;
int Datorx2=96 ;
int Datorx2_5=127;
int Datorx3=32 ;
int Datorx3_5=80;
int Datorx4=112 ;
int Datorx4_5=104;
int Datorx5=65 ;
int Datorx5_5=40;
int Datorx6=48 ;
int Datorx6_5=116;
int Datorx7=16 ;
```

```
int Datorx7_5=81;
int Datorx8=120 ;
int Datorx8_5=52;
int Datorx9=66 ;
int Datorx9_5=20;
int Datorx10=97 ;
int Datorx10_5=122;
int Datorx11=33 ;
int Datorx11_5=82;
int Datorx12=56 ;
int Datorx12_5=105;
int Datorx13=67 ;
int Datorx13_5=41;
int Datorx14=24 ;
int Datorx14_5=58;
int Datorx15=8 ;
int Datorx15_5=83;
int Datorx16=124 ;
int Datorx16_5=26;
int Datorx17=68 ;
int Datorx17_5=10;
int Datorx18=98 ;
int Datorx18_5=125;
int Datorx19=34 ;
int Datorx19_5=84;
int Datorx20=113 ;
int Datorx20_5=106;
int Datorx21=69 ;
```

```
int Datorx21_5=42;
int Datorx22=49 ;
int Datorx22_5=117;
int Datorx23=17 ;
int Datorx23_5=85;
int Datorx24=60 ;
int Datorx24_5=53;
int Datorx25=70 ;
int Datorx25_5=21;
int Datorx26=99 ;
int Datorx26_5=61;
int Datorx27=35 ;
int Datorx27_5=86;
int Datorx28=28 ;
int Datorx28_5=107;
int Datorx29=71 ;
int Datorx29_5=43;
int Datorx30=12 ;
int Datorx30_5=29;
int Datorx31=4 ;
int Datorx31_5=87;
int Datorx32=126 ;
int Datorx32_5=13;
int Datorx33=72;
int Datorx33_5=3;
int Datorx34=100;
int Datorx34_5=63;
int Datorx35=36 ;
```

```
int Datorx35_5=88;
int Datorx36=114 ;
int Datorx36_5=108;
int Datorx37=73;
int Datorx37_5=44;
int Datorx38=50 ;
int Datorx38_5=118;
int Datorx39=18 ;
int Datorx39_5=89;
int Datorx40=121 ;
int Datorx40_5=54;
int Datorx41=74;
int Datorx41_5=22;
int Datorx42=101 ;
int Datorx42_5=123;
int Datorx43=37;
int Datorx43_5=90;
int Datorx44=57 ;
int Datorx44_5=109;
int Datorx45=75 ;
int Datorx45_5=45;
int Datorx46=26;
int Datorx46_5=9;
int Datorx47=62;
int Datorx47_5=76;
int Datorx48=102;
int Datorx48_5=38;
int Datorx49=115;
```

```
int Datorx49_5=77;
int Datorx50=51;
int Datorx55=79;
int Datorx60=6;
int Datorx65=92;
int Datorx70=110;
int Datorx75=46;
int Datorx80=119;
int Datorx85=93;
int Datorx90=55;
int Datorx95=23;
int Datorx100=15;
int Datorx105=94;
int Datorx110=111;
int Datorx115=47;
int Datorx120=7;
int Datorx125=95;
int Datorx130=3;
```

```
int VARIABLE=0;     // Variable que guardará el dato recibido
                    //de la medición de temperatura o humedad segun corresponda

int T=0;            // Variable que guaradará el valor que nos indicara
                    //si el dato enviado es de humedad o temperatura.

float humedad=0;    // Variable que guarda el valor de humedad
float equivalente=0; // Variable que guarda el valor de temperatura
```

```
void setup() {
Serial1.begin(9600); // Se comienza comunicación serial 1
 lcd.begin(16, 2);   // Fijar el numero de caracteres y de filas
 lcd.clear();        // Se borra lo que este en la pantalla LCD
 Serial1.flush();    // Se limpia el buffer
}

void loop() {



   if (Serial1.available()>=2)   //Sentencia que esperará a que se reciban dos datos,
                                 //el cual el primero será el dato que nos indicará si la variable
                                 //enviada es de temperatura o humedad
                                 //Y el segundo dato es el valor de la variable sensada
```

```
{T=Serial1.read();              //Guarda el valor del primer dato que nos indicará a
                                //que variable corresponde el segundo dato recibido ( temperatura o humedad)
VARIABLE=Serial1.read();        //Guarda el valor del segundo dato recibido que corresponde a una variable sensada

// Estas sentencia no son mas que nada el equivalente del dato de variable enviada
//y la que se mostrara finalmente en pantalla, cada valor tiene su equivalente. Ver tabla.
if (VARIABLE==Datorx_4_5)
{equivalente=-4.5;}
if (VARIABLE==Datorx_4_0)
{equivalente=-4;}
if (VARIABLE==Datorx_3_5)
{equivalente=-3.5;}
```

```
if (VARIABLE==Datorx_3_0)
{equivalente=-3;}
if (VARIABLE==Datorx_2_5)
{equivalente=-2.5;}
if (VARIABLE==Datorx_2_0)
{equivalente=-2;}
if (VARIABLE==Datorx_1_5)
{equivalente=-1.5;}
if (VARIABLE==Datorx_1_0)
{equivalente=-1;}
if (VARIABLE==Datorx0)
{equivalente=0;}
if (VARIABLE==Datorx0_5)
{equivalente=0.5;}
if (VARIABLE==Datorx1)
{equivalente=1;}
if (VARIABLE==Datorx1_5)
{equivalente=1.5;}
if (VARIABLE==Datorx2)
{equivalente=2;}
if (VARIABLE==Datorx2_5)
{equivalente=2.5;}
if (VARIABLE==Datorx3)
{equivalente=3;}
if (VARIABLE==Datorx3_5)
{equivalente=3.5;}
if (VARIABLE==Datorx4)
{equivalente=4;}
```

```
if (VARIABLE==Datorx4_5)
{equivalente=4.5;}
if (VARIABLE==Datorx5)
{equivalente=5;}
if (VARIABLE==Datorx5_5)
{equivalente=5.5;}
if (VARIABLE==Datorx6)
{equivalente=6;}
if (VARIABLE==Datorx6_5)
{equivalente=6.5;}
if (VARIABLE==Datorx7)
{equivalente=7;}
if (VARIABLE==Datorx7_5)
{equivalente=7.5;}
if (VARIABLE==Datorx8)
{equivalente=8;}
if (VARIABLE==Datorx8_5)
{equivalente=8.5;}
if (VARIABLE==Datorx9)
{equivalente=9;}
if (VARIABLE==Datorx9_5)
{equivalente=9.5;}
if (VARIABLE==Datorx10)
{equivalente=10;}
if (VARIABLE==Datorx10_5)
{equivalente=10.5;}
if (VARIABLE==Datorx11)
{equivalente=11;}
```

```
if (VARIABLE==Datorx11_5)
{equivalente=11.5;}
if (VARIABLE==Datorx12)
{equivalente=12;}
if (VARIABLE==Datorx12_5)
{equivalente=12.5;}
if (VARIABLE==Datorx13)
{equivalente=13;}
if (VARIABLE==Datorx13_5)
{equivalente=13.5;}
if (VARIABLE==Datorx14)
{equivalente=14;}
if (VARIABLE==Datorx14_5)
{equivalente=14.5;}
if (VARIABLE==Datorx15)
{equivalente=15;}
if (VARIABLE==Datorx15_5)
{equivalente=15.5;}
if (VARIABLE==Datorx16)
{equivalente=16;}
if (VARIABLE==Datorx16_5)
{equivalente=16.5;}
if (VARIABLE==Datorx17)
{equivalente=17;}
if (VARIABLE==Datorx17_5)
{equivalente=17.5;}
if (VARIABLE==Datorx18)
{equivalente=18;}
```

```
if (VARIABLE==Datorx18_5)
{equivalente=18.5;}
if (VARIABLE==Datorx19)
{equivalente=19;}
if (VARIABLE==Datorx19_5)
{equivalente=19.5;}
if (VARIABLE==Datorx20)
{equivalente=20;}
if (VARIABLE==Datorx20_5)
{equivalente=20.5;}
if (VARIABLE==Datorx21)
{equivalente=21;}
if (VARIABLE==Datorx21_5)
{equivalente=21.5;}
if (VARIABLE==Datorx22)
{equivalente=22;}
if (VARIABLE==Datorx22_5)
{equivalente=22.5;}
if (VARIABLE==Datorx23)
{equivalente=23;}
if (VARIABLE==Datorx23_5)
{equivalente=23.5;}
if (VARIABLE==Datorx24)
{equivalente=24;}
if (VARIABLE==Datorx24_5)
{equivalente=24.5;}
if (VARIABLE==Datorx25)
{equivalente=25;}
```

```
if (VARIABLE==Datorx25_5)
{equivalente=25.5;}
if (VARIABLE==Datorx26)
{equivalente=26;}
if (VARIABLE==Datorx26_5)
{equivalente=26.5;}
if (VARIABLE==Datorx27)
{equivalente=27;}
if (VARIABLE==Datorx27_5)
{equivalente=27.5;}
if (VARIABLE==Datorx28)
{equivalente=28;}
if (VARIABLE==Datorx28_5)
{equivalente=28.5;}
if (VARIABLE==Datorx29)
{equivalente=29;}
if (VARIABLE==Datorx29_5)
{equivalente=29.5;}
if (VARIABLE==Datorx30)
{equivalente=30;}
if (VARIABLE==Datorx30_5)
{equivalente=30.5;}
if (VARIABLE==Datorx31)
{equivalente=31;}
if (VARIABLE==Datorx31_5)
{equivalente=31.5;}
if (VARIABLE==Datorx32)
{equivalente=32;}
```

```
if (VARIABLE==Datorx32_5)
{equivalente=32.5;}
if (VARIABLE==Datorx33)
{equivalente=33;}
if (VARIABLE==Datorx33_5)
{equivalente=33.5;}
if (VARIABLE==Datorx34)
{equivalente=34;}
if (VARIABLE==Datorx34_5)
{equivalente=34.5;}
if (VARIABLE==Datorx35)
{equivalente=35;}
if (VARIABLE==Datorx35_5)
{equivalente=35.5;}
if (VARIABLE==Datorx36)
{equivalente=36;}
if (VARIABLE==Datorx36_5)
{equivalente=36.5;}
if (VARIABLE==Datorx37)
{equivalente=37;}
if (VARIABLE==Datorx37_5)
{equivalente=37.5;}
if (VARIABLE==Datorx38)
{equivalente=38;}
if (VARIABLE==Datorx38_5)
{equivalente=38.5;}
if (VARIABLE==Datorx39)
{equivalente=39;}
```

```
if (VARIABLE==Datorx39_5)
{equivalente=39.5;}
if (VARIABLE==Datorx40)
{equivalente=40;}
if (VARIABLE==Datorx40_5)
{equivalente=40.5;}
if (VARIABLE==Datorx41)
{equivalente=41;}
if (VARIABLE==Datorx41_5)
{equivalente=41.5;}
if (VARIABLE==Datorx42)
{equivalente=42;}
if (VARIABLE==Datorx42_5)
{equivalente=42.5;}
if (VARIABLE==Datorx43)
{equivalente=43;}
if (VARIABLE==Datorx43_5)
{equivalente=43.5;}
if (VARIABLE==Datorx44)
{equivalente=44;}
if (VARIABLE==Datorx44_5)
{equivalente=44.5;}
if (VARIABLE==Datorx45)
{equivalente=45;}
if (VARIABLE==Datorx45_5)
{equivalente=45.5;}
if (VARIABLE==Datorx46)
{equivalente=46;}
```

```
if (VARIABLE==Datorx46_5)
{equivalente=46.5;}
if (VARIABLE==Datorx47)
{equivalente=47;}
if (VARIABLE==Datorx47_5)
{equivalente=47.5;}
if (VARIABLE==Datorx48)
{equivalente=48;}
if (VARIABLE==Datorx48_5)
{equivalente=48.5;}
if (VARIABLE==Datorx49)
{equivalente=49;}
if (VARIABLE==Datorx49_5)
{equivalente=49.5;}
if (VARIABLE==Datorx50)
{equivalente=50;}
if (VARIABLE==Datorx55)
{equivalente=55;}
if (VARIABLE==Datorx60)
{equivalente=60;}
if (VARIABLE==Datorx65)
{equivalente=65;}
if (VARIABLE==Datorx70)
{equivalente=70;}
if (VARIABLE==Datorx75)
{equivalente=75;}
if (VARIABLE==Datorx80)
{equivalente=80;}
```

```
if (VARIABLE==Datorx85)
{equivalente=85;}
if (VARIABLE==Datorx90)
{equivalente=90;}
if (VARIABLE==Datorx95)
{equivalente=95;}
if (VARIABLE==Datorx100)
{equivalente=100;}
if (VARIABLE==Datorx105)
{equivalente=105;}
if (VARIABLE==Datorx110)
{equivalente=110;}
if (VARIABLE==Datorx115)
{equivalente=115;}
if (VARIABLE==Datorx120)
{equivalente=120;}
if (VARIABLE==Datorx125)
{equivalente=125;}
if (VARIABLE==Datorx130)
{equivalente=130;}
```

```
humedad=equivalente;}       // Aqui guardamos el segundo valor enviado,
                            //en la variable  temperatura y humedad, para
                            //luego definir a que variable corresponde el dato.

if(T==14)                   // Si el dato del primer valor enviado es 14 (ver tabala de valores equivalentes)
                            //el segundo dato sera de la variable temperatura, de lo contrario sera humedad.
{                           // Y procedemos a mostrarlo en pantalla
```

```
lcd. setCursor ( 0 , 0 ) ;  // Fijamos la fila  y columna desde donde se emepzara a
                            //escribir el dato de la variable temperatura
lcd.print ( equivalente);   // Mostramos en pantalla el valor final de la variable  sensada de temperatura
lcd.print (" C");           // Colocamos en pantalla el simbolo de temperatura en grados celsius
delay(1990);                // Damos un tiempo antes de empezar a recibir el  valor de la variable humedad
}
else {              //Sentencia en caso que la variable("T") que define si el dato corresponde a humedad o
                    //temperatura no fuera 14, aqui empezariamos a mostrar el valor la variable humedad
```

```
  lcd. setCursor ( 0 , 1 ) ; // Fijamos el cursor de la pantalla LCD en la columna 0 y fila 1, esta vez
                             //nos corrimos para no borrar el dato de temperatura mostrada en pantalla,
                             //para asi mostrar los dos valores
lcd.print ( humedad);       // Mostramos en pantalla el valor final de la variable  sensada de humedad
lcd.print (" %");           // Colocamos en pantalla el simbolo de humedad en porcentaje
delay (1990);               // Damos un tiempo antes de empezar a recibir el  valor de la variable temperatura otra vez
}
}
```