29-11-2016

MONITORIZACIÓN DE LA CALIDAD DE AGUAS PLUVIALES VERTIDAS A CAUCE PÚBLICO EN ENTORNOS INDUSTRIALES MEDIANTE TECNOLOGÍA LOW COST

Autor: Inmaculada González Ruiz.
Tutor: Pablo Matute Martín
Departamento: Ingeniería Química y Ambiental

INDICE.

1 INTRODUCCIÓN

A día de hoy, la **monitorización** resulta imprescindible para poder gestionar sistemas y registrar datos (de todo tipo) de una forma **segura** y **eficiente**. Para poder llevar a cabo esta misión, desde hace tiempo existe una amplia gama de sensores en el mercado dependiendo de la magnitud a medir y de las prestaciones que se requieran.

El estudio de las condiciones ambientales y calidad del agua se lleva realizando desde el pasado siglo. Con la aparición de equipos de bajo coste que pueden llevar a cabo esta tarea de manera sencilla y sin muchos recursos, es necesario conocer sus aplicaciones, prestaciones y limitaciones que pueden tener para elegir adecuadamente cuál es apto y cuál no para cierto sistema.

Adicionalmente, la inclusión de este tipo de sensores en un **sistema aislado** nos da la posibilidad de poder monitorizarlo de forma remota, sin tener acceso físico a él y, con ello, reducir costes de inspección y control.

La llegada de soluciones "***hardware low cost y de código abierto***" ha hecho posible montar, **automatizar** y configurar todo un sistema bajo la dirección de una única plataforma, de forma sencilla y **eficiente**. La solución que se plantea en este documento va encaminada justamente en esa dirección.

Gracias al gran soporte de este tipo de plataformas por parte de los usuarios, es posible encontrar todo tipo de proyectos en la Red; desde los más sencillos a los que más complejidad entraña. Es, gracias a esta comunidad, que los dispositivos de "hardware" libre están de moda en estos días, facilitando enormemente el trabajo en aplicaciones rutinarias, procesos automatizados, monitorización, etcétera.

Por otro lado el poder combinar el medio de comunicación de **radiofrecuencia**, una tecnología que ofrece la posibilidad de implementar un sistema **sin cables**, de **largo alcance** y **seguro**, permitiendo así **independencia** del conjunto de medios físicos con el hardware de bajo coste y la nuevas plataformas basadas en la tecnología IoT, permiten desarrollar números prototipos que nos dan acceso a parámetros ambientales en diversas disciplinas, como puede ser la calidad del agua o la emisiones al aire.

2 CAPITULO 1: CALIDAD DEL AGUA PLUVIAL CONTAMINADA EN ENTORNOS INDUSTRIALES.

2.1 ANTECEDENTES.

Las aguas pluviales recogidas en diferentes entornos industriales procedentes de diversas industrias como la minera, vertederos, fertilizantes…, suelen estar contaminadas con metales, fosfatos, nitratos, DQO… Cuando estas aguas llegan al cauce de los ríos o lagos, provocan problemas serios de contaminación ambiental.

Los iones disueltos de los metales pesados, como el plomo, cadmio y mercurio, son muy tóxicos y acumulables por los organismos que los absorben, los cuales a su vez son fuente de contaminación de las cadenas alimenticias al ser ingeridos por alguno de sus eslabones.

La contaminación de las aguas pluviales con fosfatos y nitratos cuando llegan a las rías, lagos, ríos…., provocan un tipo de contaminación química conocida como eutrofización, esto hecho se da cuando hay un aporte excesivo de nutrientes a un ecosistema acuático, el cual queda severamente afectado por ello. El resultado son ecosistemas con una biodiversidad reducida, con las especies oportunistas ocupando nichos previamente ocupados por otras especies.

2.2 OBJETIVO.

2.2.1 Objetivo general.

Obtener información en tiempo real de diferentes parámetros relevantes indicadores de la calidad del agua pluvial en entornos industriales, implementando para ello un prototipo basados en tecnologías libres que permitan el monitoreo de dichas agua y su posible actuación en el control de equipos que evacuen o impidan el vertido de esa agua contaminada a un cauce público, como podría ser una ría.

2.2.2 Objetivo específico.

- ★ Diseño y construcción de un prototipo que contenga un grupo de sensores para medir los parámetros principales, siendo estos parámetros:
    1. Temperatura
    2. pH
    3. Conductividad.
    4. Turbidez
    5. Pluviometría.
- ★ Diseño y desarrollo del código en la plataforma Arduino para la monitorización de los parámetros seleccionados.
- ★ Diseño y desarrollo de un interfaz realizado en la aplicación My Scada para la visualización de los datos en PC, SamrtPhone. IPad…..
- ★ Configuración de la plataforma ThinkSpeak como servidor de datos y visualización de las mismas directamente desde Internet.

2.3 DESCRIPCIÓN DEL PROBLEMA.

En la actualidad existen números entornos industriales tratan sus agua de lluvia en tanques de tormenta o bien las derivan directamente a colectores conectados a un cauce público.

Asimismo, los materiales sólidos de desechos, restos de materia prima dispersa por las zonas de operación, son generalmente acumulados por las superficies del terreno, y si no está propiamente dispuesto, se exponen a lluvias o corrientes de agua superficiales cercanas, siendo lavado por el agua de la lluvia y los contaminantes son disueltos o arrastrados por el agua.

Con la aparición de las lluvias estas aguas son recogidas a través de los imbornales que finalmente derivan en diferentes tanques de tormentas situados estratégicamente en las plantas. Cuando estas aguas alcanzan la cota de vertido, dichas aguas van a parar al cauce público.

En la foto siguiente puede observarse un entorno industrial donde se visualiza un suelo contaminado de concentrado de cobre, dicho concentrado cuando llueve es recogido en los imbornales que dirigen el agua hasta el tanque de tormenta, que finalmente ante el aumento del nivel del tanque el agua contaminada deriva en el cauce público.

En otros entornos industriales como son los centros de gestión de residuos de sólidos urbanos existen la problemática de la contaminación difusa debido a que los residuos con el viento y el trasiego de camiones se van esparciendo por el recinto, además en los parques de maduración al aire libre, se forma lixiviado que cuando llueve, puede llegar a las cunetas de pluviales y pueden salir del centro si estos no son bien gestionados. Además con la lluvia puede haber problemas de contaminación en los frentes de vertidos.

Por ello a la salida del agua de pluviales es interesante un sensor de monitorización para avisar en caso de subida de niveles de conductividad y pH para poder actuar de forma adecuada.
En las siguientes fotos puede observar esta problemática.

MONITORIZACIÓN DE LA CALIDAD DE AGUAS PLUVIALES VERTIDAS A CAUCE PÚBLICO EN ENTORNOS INDUSTRIALES MEDIANTE TECNOLOGÍA LOW COST

Autor: Inmaculada González

2.4 CALIDAD DE LAS AGUAS PLUVIALES.

Las aguas naturales, al estar en contacto con diferentes agentes (aire, suelo, vegetación, subsuelo, etc.), incorporan parte de los mismos por disolución o arrastre.

- ★ Esto hace que las aguas dulces presenten un elevado número de sustancias en su composición química natural.
- ★ Entre los compuestos más comunes que se pueden encontrar en las aguas dulces están:
  - como constituyentes mayoritarios: los carbonatos, bicarbonatos, sulfatos, cloruros y nitratos.
  - como constituyentes minoritarios: los fosfatos y silicatos, metales como elementos traza y gases disueltos como oxígeno, nitrógeno y dióxido de carbono.
- ★ El agua de lluvia presenta
  - los cationes: Na^+, K^+, Ca^{2+}, Mg^{2+}
  - los aniones: HCO_3^-, Cl^-, Br^-, I^-, SO_4^{2-}, NO_3^-, PO_4^{3-}
  - y dióxido de carbono, oxígeno, ozono, nitrógeno, argón, etc.

La composición química natural de las aguas puede verse alterada por actividades humanas: agrícolas, industriales, etc., incorporando sustancias de diferente naturaleza debido al paso de las aguas por terrenos tratados con productos agroquímicos o contaminados.

- ★ Estas incorporaciones ocasionan la degradación de la calidad del agua provocando diferentes efectos negativos como:
  - la modificación de los ecosistemas acuáticos
  - la destrucción de los recursos hidráulicos
  - riesgos para la salud
  - incremento del coste del tratamiento del agua para su uso
  - daño en instalaciones (incrustaciones, corrosiones, etc.)

- ★ Las aguas contaminadas presentan diversos compuestos en función de su procedencia: pesticidas, tenso activos, fenoles, aceites y grasas, metales pesados, etc.
- ★ La composición específica de un agua determinada influye en propiedades físicas tales como densidad, tensión de vapor, viscosidad, conductividad, etc.
- ★ La progresiva contaminación cambia sustancialmente sus propiedades.
- ★ Las filtraciones, los vertidos han dado lugar a que, a veces, el agua pluvial sea agua contaminada o agua de proceso. Esto ha originado la necesidad de utilizar parámetros de control.
- ★ Algunos de los parámetros de control son los seleccionados por el presente proyecto para su monitorización.

2.4.1 Parámetros seleccionados para la detección de vertido de aguas pluviales.

En esta sección se va a resumir la información más relevante con el objetivo de realizar la selección de los parámetros a detectar por el sistema.

Como dato determinante para la elección de parámetros para determinar la existencia de un vertido, se ha usado el coste del sensor, ya que, el objetivo del sistema es evitar el vertido de agua de lluvia contaminada a cauce público mediante equipos de bajo coste.

Evidentemente el prototipo puede complementarse con todos aquellos sensores que la zona a monitorizar por sus características lo requieran. Estos sensores complementarios podrían ser la detección de cobre, zinc, nitratos…, son sensores de bajo coste en comparación con los señores tradicionales, pero que para el diseño y construcción del presente prototipo suponen coste elevado para adjuntar al presente TFM.

Por esta razón se ha estudiado diferentes parámetros que nos pueden aportar información significativa de la calidad del agua de lluvia con un presupuesto de bajo coste.

Las medidas consideradas para el prototipo han sido:

- pH: Nos indica la acidez o alcalinidad del agua.
- Conductividad. Hace referencia a la salinidad del agua.
- Turbidez. Nos indica orienta sobre los sólidos totales en suspensión en el agua

- Temperatura. Sirve como dato informativo y es necesaria para medición de conductividad y pH.
- Pluviometría. Nos permite detectar la cantidad de lluvia que cae en un lugar y en un espacio de tiempo.

PH.

En primer lugar, se recuerda que se preseleccionaron el PH y temperatura, pH porque es primordial para detectar la alcalinidad o acidez del agua y el de temperatura porque se necesita una compensación de la misma para la medición de pH.

El pH se define como:

$PH = \log 1/ [H+] = -\log [H+]$

- ★ Es una propiedad que afecta a muchas reacciones químicas y biológicas.
- ★ El valor del pH compatible con la vida piscícola está comprendido entre 5 y 9. Para la mayoría de las especies acuáticas, la zona de pH favorable se sitúa entre 6.0 y 7.2. Fuera de este rango no es posible la vida como consecuencia de la desnaturalización de las proteínas.
- ★ La alcalinidad es la suma total de los componentes en el agua que tienden a elevar el pH (bases fuertes y sales de bases fuertes y ácidos débiles).
- ★ La acidez es la suma de componentes que implican un descenso de pH (dióxido de carbono, ácidos minerales, ácidos poco disociados, sales de ácidos fuertes y bases débiles).
- ★ Ambas, controlan la capacidad de tamponamiento del agua (para neutralizar variaciones de pH provocadas por la adición de ácidos o bases).
- ★ El principal sistema regulador del pH en aguas naturales es el sistema carbonato (dióxido de carbono, ion bicarbonato y ácido carbónico).

Conductividad.

En los parámetros seleccionados para detectar un vertido, se considera como medida significativa la conductividad.

Es la medida de la capacidad del agua para transportar la corriente eléctrica y permite conocer la concentración de especies iónicas presentes en ella.

- ★ La contribución de cada especie iónica a la conductividad es diferente por lo que su medida da un valor que no está relacionado con el número total de iones en solución. Depende también de la temperatura.
- ★ Está relacionada con el residuo fijo por la expresión
- ★ conductividad (µS/cm) x f = residuo fijo (mg/L); El valor de f varía entre 0.55 y 0.9.
- ★ a salinidad de las soluciones de sistemas de suelo, agua de riego o fertilizantes es un parámetro importante que afecta a la zona de las raíces de las plantas ambiente. Cualquiera de estos puede descomponer en factores-tener un efecto significativo sobre el crecimiento vegetal y la calidad.

Turbidez

Es una medida de la dispersión de la luz por el agua por la presencia de materiales suspendidos coloidales y/o articulados.

- ★ La materia suspendida puede indicar un cambio en la calidad del agua y/o la presencia de sustancias inorgánicas finamente divididas o de materiales orgánicos.
- ★ La turbidez es un factor ambiental importante ya que la actividad fotosintética depende en gran medida de la penetración de la luz.
- ★ La turbidez constituye un obstáculo para la eficacia de los tratamientos de desinfección.
- ★ La transparencia del agua es muy importante en las de aguas potables y en el caso de industrias que producen materiales destinados al consumo humano

Temperatura.

Además de ser un parámetro necesario para la medida de conductividad y pH, la temperatura resulta relevante debido a:

- ★ Temperaturas elevadas implican la aceleración de la putrefacción, con lo que aumenta la DBO y disminuye el oxígeno disuelto.
- ★ La temperatura tiene influencia en el equilibrio entre el nitrógeno amoniacal ionizado y el amoniaco, con la temperatura aumenta la saturación del oxígeno y el incremento de la misma puede afectar a la vida acuática en algunas etapas de sus ciclos vitales

3 CAPITULO 2: DISEÑO DEL SISTEMA DE DETECCIÓN.

3.1 DESCRIPCIÓN DE SENSORES UTILIZADOS.

Se presentan los diferentes periféricos a instalar en el prototipo para la monitorización de aguas pluviales. Para ello se establece en este capítulo una serie de apartados considerados relevantes para disponer de toda la información necesaria de cada sensor, así con su funcionalidad dentro del prototipo diseñado.

3.2 UBICACIÓN DE SENSORES.

Para la medición de los parámetros del agua a monitorizar en Tanques de tormenta los sensores, tenemos se instalarían en un lugar externo al tanque de tormenta.

Como nos interesa medir la calidad del agua antes de que el agua llegue a la cota de aliviadero del tanque de tormenta, acondicionaremos una muestra en continuo en un cubeto donde quedarían instalados los sensores.

En realidad con esta ubicación, siempre estamos midiendo los valores del tanque tanto si está vertiendo a cauce público como si no lo está haciendo. Para ello se instala una pequeña bomba que eleva el agua hasta el cubeto donde están ubicados los sensores. Existe detector de nivel tipo hidromel o boya a la altura del vertedero para saber cuándo se está vertiendo el agua a cauce público.
Como el tanque de tormentas tiene varios metros de profundidad con este sistema evitamos que se inunden los sensores y queden fuera de uso.

3.3 INDICE DE INSTRUMENTOS.

En la lista siguiente se realiza una todos los instrumentos que intervienen en el sistema, con la información necesaria para la correcta interpretación de su función y las características generales.

Este índice de instrumentos recopila la información más importante asociada a cada instrumento, desde marca y modelo del sensor hasta su rango de medida.

TAG	DESCRIPCIÓN	SERVICIO	UBICACIÓN	MARCA	MÓDELO	RANGO MEDIDA	TIPO ENTRADA	TENSIÓN
TT01-AIT$_{[PH]}$-01	Transmisor e indicador de pH	Indicar y transmitir pH	Arduino	DFROBOT	SEN 0161	0-14	Analógica	5 VDC
TT01-AE$_{[pH]}$-01	Sensor de pH	Medición pH	Tanque Tormenta	DFROBOT				
TT01-AIT$_{[Co]}$-01	Transmisor e indicador de Co	Indicar y transmitir Co	Arduino	DFROBOT	DFR0300	1 ms / cm - 20 ms / cm	Analógica	5 VDC
TT01-AE$_{[Co]}$-01	Sensor de Conductividad	Medición Co	Tanque Tormenta	DFROBOT				
TT01-AIT$_{[Tb]}$-01	Transmisor e indicador de Turbidez	Indicar y transmitir Tb	Arduino	DFROBOT	SEN 0189		Analógica	5 VDC

TAG	DESCRIPCIÓN	SERVICIO	UBICACIÓN	MARCA	MÓDELO	RANGO MEDIDA	TIPO ENTRADA	TENSIÓN
TT01-AE$_{[Tb]}$-01	Sensor de turbidez	Medición Tb	Tanque Tormenta	DFROBOT				
TT01-TIT-01	Transmisor e indicador de temperatura	Indicar y transmitir Temperatura	Arduino	ATLAS SCIENTIFIC	DS18B20	± 0,5 ° C de -10 ° C a + 85 ° C	Digital	5 VDC
TT01-TE-01	Sensor de temperatura	Medición Temperatura	Tanque Tormenta	ATLAS SCIENTIFIC				
TT01-QI-01	Detector de cantidad de lluvia	Medición de lluvia por área	Intemperie		Peysanet		Digital	5 VDC

3.4 SISTEMA DE DETECCIÓN DE PH.

Sensor de pH SEN 0161.

PH meter (SKU: SEN0161)

3.4.1 Descripción.

El sensor de Potencial de Hidrógeno (pH) es un transductor que permite conocer el pH de una solución, esto lo realiza a través de un método electroquímico que utiliza una membrana de vidrio que separa dos sustancias con diferentes cantidad de, el sensor es un elemento pasivo que genera una pequeña cantidad de corriente de acuerdo al nivel de pH que se encuentre en el medio ambiente.

El sensor de pH seleccionado permite una rápida solución para diseños de bajo costo sin que se sacrifique la operatividad del diseño, además de ser un sistema embebido ya que el sensor consta como hemos indicado en el párrafo anterior de una sonda de pH, que es un elemento pasivo que detecta una pequeña corriente eléctrica generada por la actividad de los iones de Hidrógeno.

Consta de tres elementos:

- Sensor de pH.
- Placa de conversión a Arduino.
- Cable de conexionado entre la placa Arduino y el conversor.

El sensor tiene un LED que funciona como el indicador de alimentación, un conector BNC para unirse a la interfaz del sensor PH2.0. Sólo puede conectar el sensor de pH con conector BNC y conectar la interfaz PH2.0 a cualquier entrada analógica del controlador Arduino para leer el valor del pH.

Paso para usar el medidor de pH

3.4.2 Características.

VOLTAJE DE FUNCIONAMIENTO:	5V DC.
CORRIENTE DE FUNCIONAMIENTO:	
TIEMPO DE RESPUESTA:	≤ 1min
RANGO DE MEDICIÓN	0-14
PRECISIÓN	± 0.1pH (25 ℃)
POTENCIOMETRO	Potenciómetro de ajuste de ganancia
INIDCADORES	Indicador de encendido LED
MEDICIÓN DE LA TEMPERATURA	0-60 ℃
CONECTOR SENSOR	Conector BNC
CONECTOR INTERFAZ PH 2.0	Conector de 3 pines
DIMENSIONES DEL ADAPTADOR:	43mm × 32mm

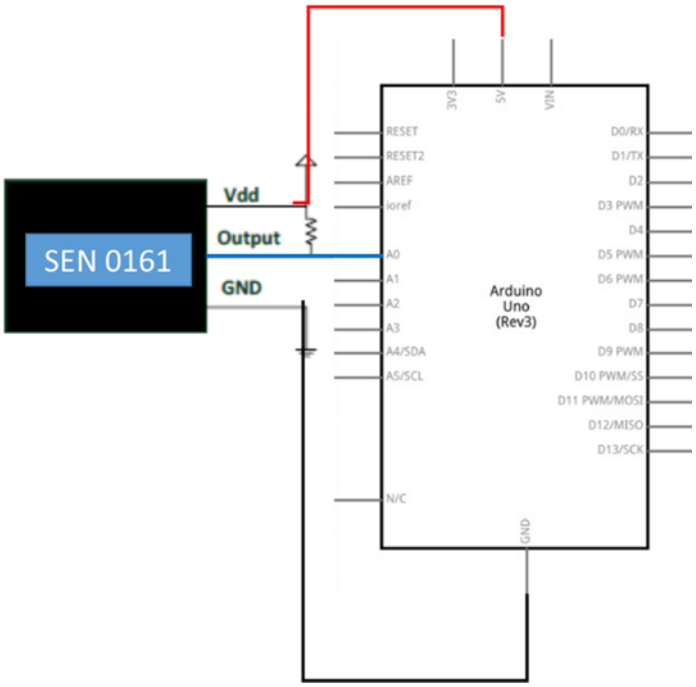
Característica electrodo.

La salida del electrodo del pH es Millivolts, y el valor de pH de la relación se muestra como sigue (25 ℃):

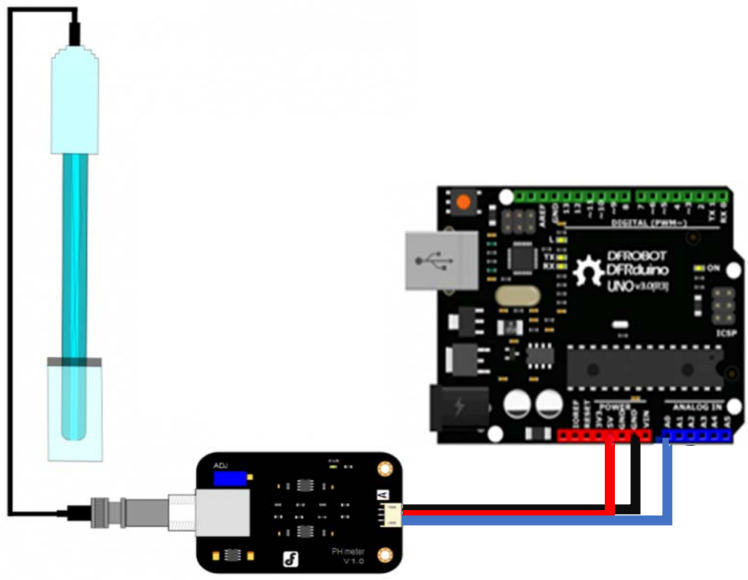
VOLTAGE (mV)	pH value	VOLTAGE (mV)	pH value
414.12	0.00	-414.12	14.00
354.96	1.00	-354.96	13.00
295.80	2.00	-295.80	12.00
236.64	3.00	-236.64	11.00
177.48	4.00	-177.48	10.00
118.32	5.00	-118.32	9.00
59.16	6.00	-59.16	8.00
0.00	7.00	0.00	7.00

3.4.3 Diagrama de bloque

Para lograr el objetivo de monitorear en tiempo real la conductividad, en la figura 3.4.3 se presenta el diagrama a bloques propuesto de acoplamiento entre el sensor **SEN0161** y la placa Arduino UNO R3 a través del pin analógico número **A0** con una comunicación 1-wire.



3.4.4 Esquema de conexionado.

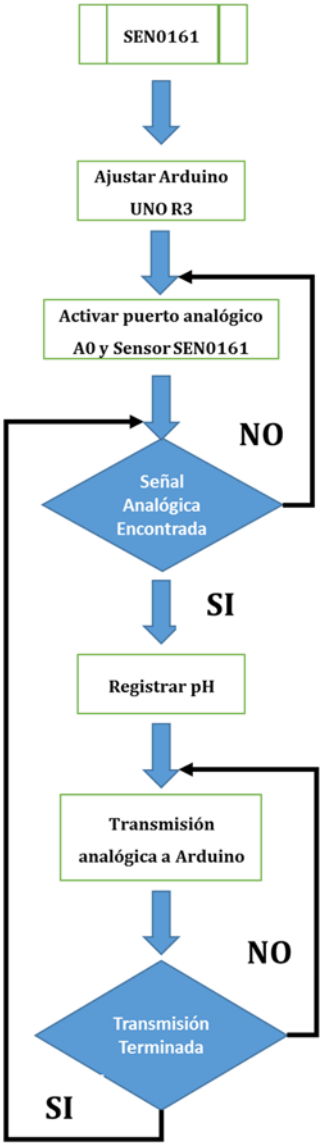


Autor: Inmaculada González

3.4.5 Diagrama de flujo

A continuación se muestra el diagrama de flujo para obtener la variable de pH.

Se ajusta la placa Arduino UNO R3 para la recepción de datos a través del puerto analógico A0, activando a su vez el sensor SEN0161. Al detectar la señal analógica, comienza a registrar la variable de pH y manda dichos datos obtenidos por su pin analógico a la placa Arduino UNO R3. Si no es encontrada la señal, no se registran, ni transmite los datos. El programa se ejecuta de manera cíclica.



3.4.6 Esquema Placa Sensor.

3.5 SISTEMA DE DETECCIÓN DE CONDUCTIVIDAD.

Sensor de Conductividad DFR0300.

3.5.1 Descripción.

El sensor de Conductividad DFR0300 viene suministrado con un sensor de temperatura descrito en el punto 3.9, dicho sensor es necesario para la medida de conductividad porque se requiere una compensación de la temperatura para realizar correctamente la medida de conductividad.

Se ha seleccionado este medidor porque está especialmente diseñado para los controladores de Arduino y tiene una función de características simples. Dispone de fáciles conexiones y posee un software de fácil uso para la calibración del sensor. Además para facilitar la configuración del sensor, el interfaz Gravity dispone de un sistema plug & play.

Lista de componentes:

- Un Electrodo de conductividad con conector BNC.
- Tarjeta de circuito.
- Cable analógico.
- Un sensor de temperatura modelo DS18B20 (impermeable).
- Terminal adaptador Del sensor.
- Cable digital.
- Solución estándar de conductividad (1413us/cm and 12.88ms/cm).

3.5.2 Principio de medida.

En primer lugar, nos encontramos con el chip U3B. Consiste en el circuito escalado inverso. La función de transferencia es Vo = R10 / R * Vi. R10 es una resistencia de retroalimentación y su valor es 820ohm de acuerdo con el esquema. R es la resistencia cuando el electrodo se inserta en solución acuosa. Su valor está relacionado con la conductividad de la solución acuosa. R10 / R se llama ampliación. Cuando se cambia el R, también se cambia la ampliación, por lo que el Vo cambiado. Entonces Vo está relacionado con R. A la derecha del circuito escalado inverso, hay un circuito de valor absoluto. Su función de transferencia es Vo = | vi |. Arduino muestrea la salida del circuito de valor absoluto para calcular la conductividad.
A continuación se analiza el principio de calibración.
La definición de resistencia es:

$$R = \rho \frac{L}{A}$$

ρ: La resistividad
L: La longitud del conductor
A: La sección del conductor

Para el electrodo de conductividad, L es la separación entre dos hojas conductoras, A es el área de la lámina conductora.

Definición de conductividad:

$$\kappa = \frac{1}{\rho}$$

Igualando las ecuaciones tenemos el resultado

$$\kappa = \frac{1}{\mathrm{R}} \bullet \frac{L}{A}$$

Donde

1/R es la conductancia

Y a L/A se le denomina constante de Vessel Q

La función de transferencia de la medida del circuito es

$$Vout = \frac{\mathrm{R10}}{R} \times |Vin|$$

R es la resistencia del electrodo en el medio acuoso

En conclusión tenemos la siguiente formula:

$$\kappa = \frac{\mathrm{Q}}{R10 \bullet |Vin|} \times Vout$$

Q es la constante de Vessel. Es una constante y diferente para cada electrodo. En el esquema, R10 es 820Ω.

| Vin | Es también una constante dependen del circuito generador de señal. Su valor es de unos 200mV.

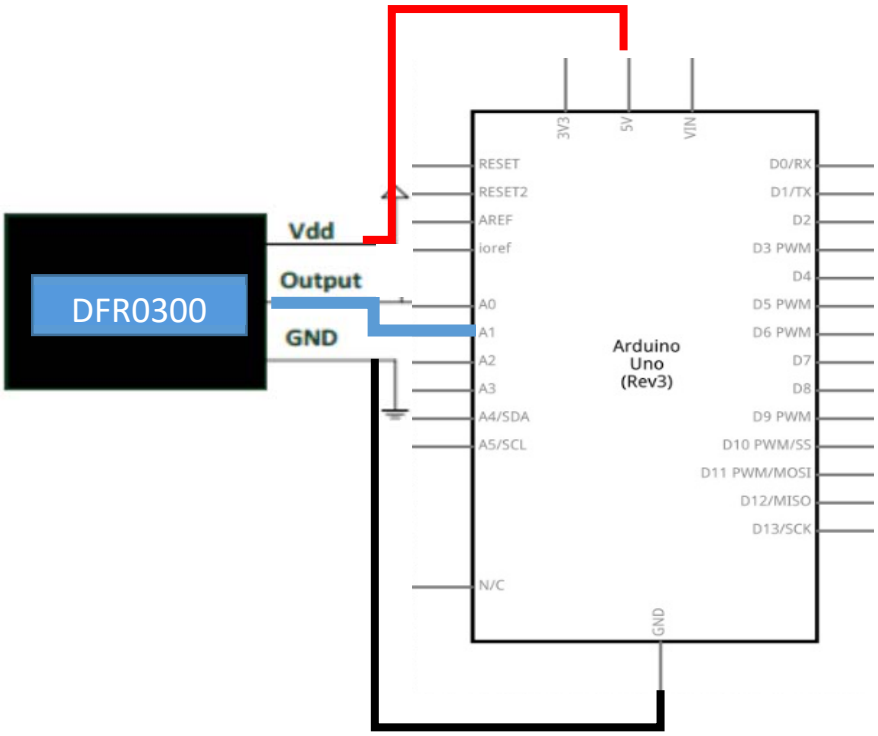
Así que podemos ver, la conductividad es lineal con la tensión de salida.

3.5.3 Características.

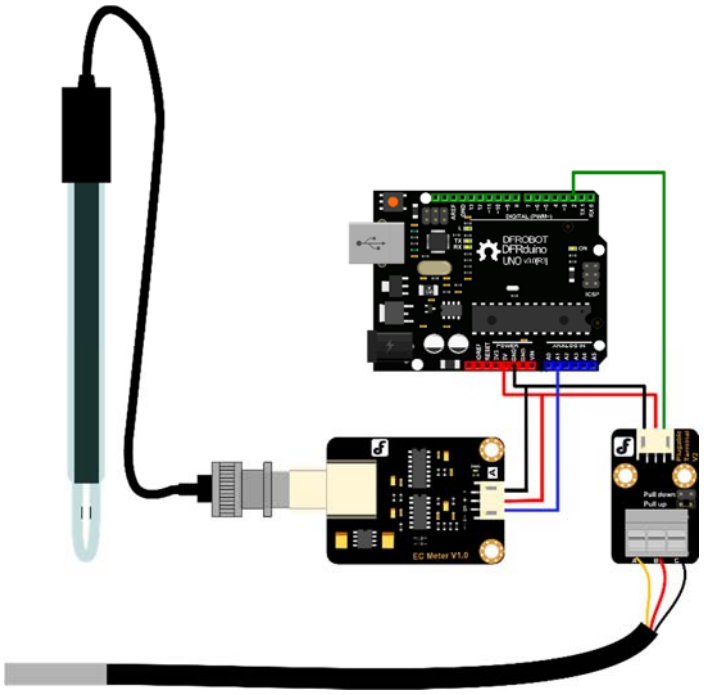
VOLTAJE DE FUNCIONAMIENTO:	5V DC.
TAMAÑO DE PCB:	45x32 mm
INTERVALO DE MEDIDA	1 ms / cm - 20 ms / cm
RANGO DE MEDICIÓN	0-14
TEMPERATURA DE FUNCIONAMIENTO	5-40 ℃
PRECISIÓN	<± 10% F.S (usando el ADC de 10 bits Arduino)
ELECTRODO	electrodo constante K = 1, conector BNC
INDICADOR DE ENCENDIDO	LED
LONGITUD DEL CABLE DEL ELECTRODO	unos 60 cm
CONECTOR SENSOR	Conector BNC
CONECTOR INTERFAZ PH 2.0	Conector de 3 pines
DIMENSIONES DEL ADAPTADOR:	43mm × 32mm

3.5.4 Diagrama de bloque

Para lograr el objetivo de monitorear en tiempo real la conductividad, en la figura 3.6.2 se presenta el diagrama a bloques propuesto de acoplamiento entre el sensor **DFR0300** y la placa Arduino UNO R3 a través del pin analógico número **A1,** la medida de temperatura será enviada por una comunicación 1-wire (como se explica más adelante).



3.5.5 Esquema de conexionado



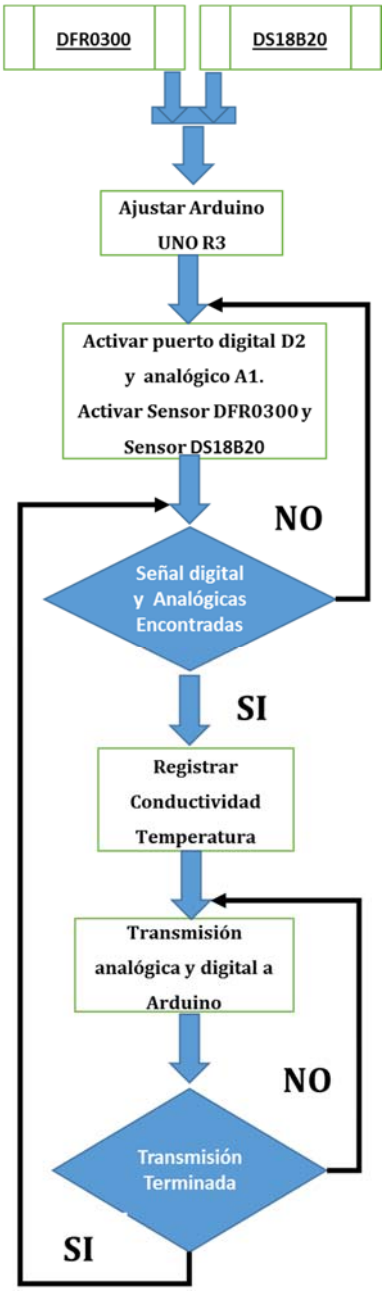
Autor: Inmaculada González

3.5.6 Diagrama de flujo

En la figura 3.6.5 se muestra el diagrama de flujo para obtener la variable de conductividad.
Inicialmente debe calibrase el sensor de conductividad.

Se ajusta la placa Arduino UNO R3 para la recepción de datos del sensor de temperatura mediante el puerto digital 2 (D2), activando a su vez el sensor DS18B20. Al detectar la señal digital, comienza a registrar la variable de temperatura y manda dichos datos obtenidos por su pin digital a la placa Arduino UNO R3 para la obtención de la conductividad.

Por otro lado, se ajusta la placa Arduino UNO R3 para la recepción de datos a través del puerto analógico A1, activando a su vez el sensor **DFR0300**. Al detectar la señal analógica, comienza a registrar la variable de conductividad y manda dichos datos obtenidos por su pin analógico a la placa Arduino UNO R3, éste junto con los datos de la temperatura calcula la medida de la conductividad.
Si las señales no son encontradas, no se registran, ni transmite los datos. El programa se ejecuta de manera cíclica.



3.5.7 Esquema Placa Sensor.

3.6 SISTEMA DE DETECCIÓN DE TURBIDEZ.

Sensor de Turbidez SKU:SEN0189.

Gravity: Analog Turbidity Sensor para Arduino.

3.6.1 Descripción.

El sensor de Turbidez SKU: SEN0189
El sensor de turbidez detecta la calidad del agua midiendo el nivel de turbidez. Es capaz de detectar partículas en suspensión en el agua midiendo la transmitancia de la luz y la velocidad de dispersión que cambia con la cantidad de sólidos suspendidos totales (TSS) en agua. A medida que el TTS aumenta, el nivel de turbidez líquido aumenta.

Este sensor tiene los dos modos de señal de salida, tanto analógica como digital. Se puede seleccionar el modo según la MCU, ya que, el umbral es ajustable en el modo de señal digital.

Este sensor de turbidez puede usarse para medir la calidad del agua en ríos y arroyos, mediciones de aguas residuales y efluentes, investigación en transporte de sedimentos y mediciones de laboratorio.

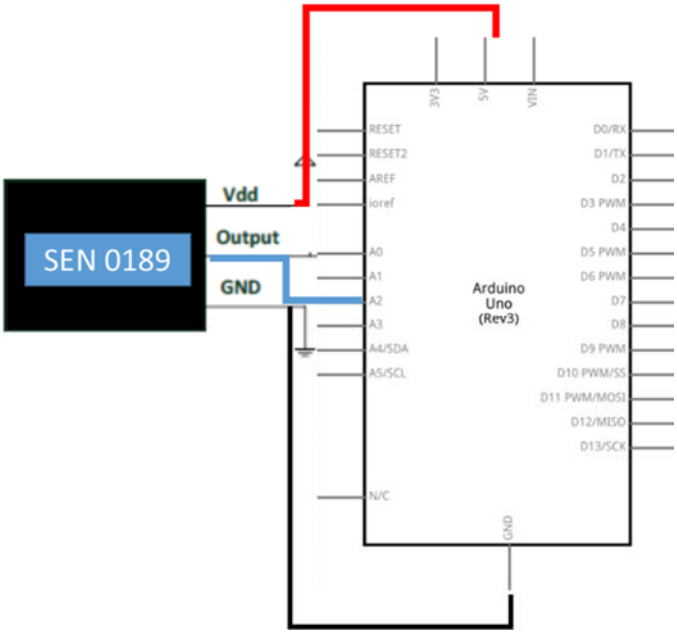
Por su funcionalidad resulta un parámetro de medida muy significativo en la detección de vertido, ya que, su valor está relacionado directamente con la cantidad de sólidos totales en suspensión en la disolución acuosa.

3.6.2 Características.

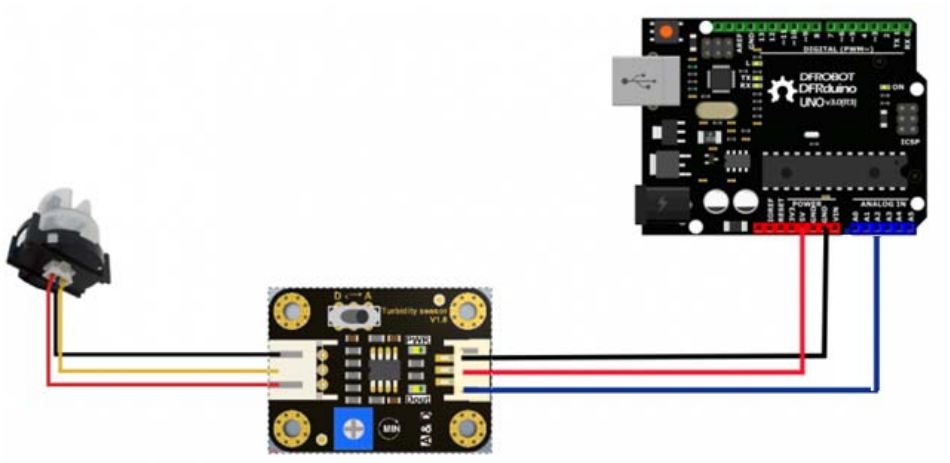
VOLTAJE DE FUNCIONAMIENTO:	5V DC.
CORRIENTE DE FUNCIONAMIENTO:	40mA (MAX).
TIEMPO DE RESPUESTA:	<500ms.
RESISTENCIA DE AISLAMIENTO:	100M (Min).
MÉTODO DE SALIDA:	Analógico.
SALIDA ANALÓGICA:	0-4.5V.
SALIDA DIGITAL:	Señal de nivel alto / bajo (puede ajustar el valor de umbral ajustando el potenciómetro).
TEMPERATURA DE FUNCIONAMIENTO:	5 ℃ ~ 90 ℃.
TEMPERATURA DE ALMACENAMIENTO:	-10 ℃ ~ 90 ℃.
PESO:	30g.
DIMENSIONES DEL ADAPTADOR:	38mm * 28mm * 10m m / 1.5inches * 1.1inches * 0.4inches.

3.6.3 Diagrama de bloque

Para lograr el objetivo de monitorear en tiempo real la conductividad, en la figura 3.1.6.3 se presenta el diagrama a bloques propuesto de acoplamiento entre el sensor **SKU: SEN0189** y la placa Arduino UNO R3 a través del pin analógico número A2 con una comunicación 1-wire.



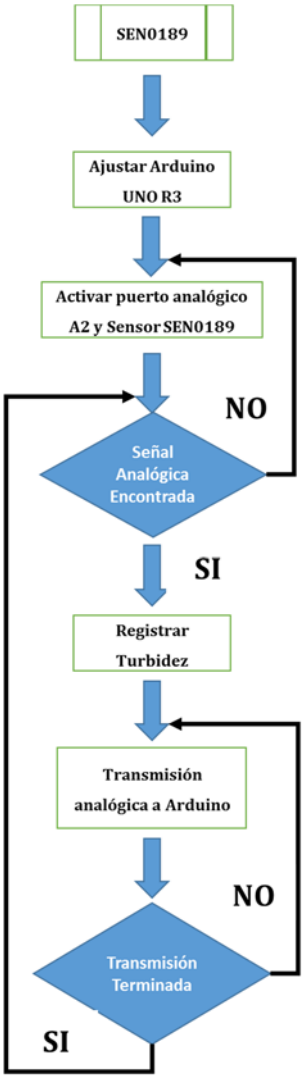
3.6.4 Esquema de conexionado



3.6.5 Diagrama de flujo

A continuación se muestra el diagrama de flujo para obtener la variable de turbidez.

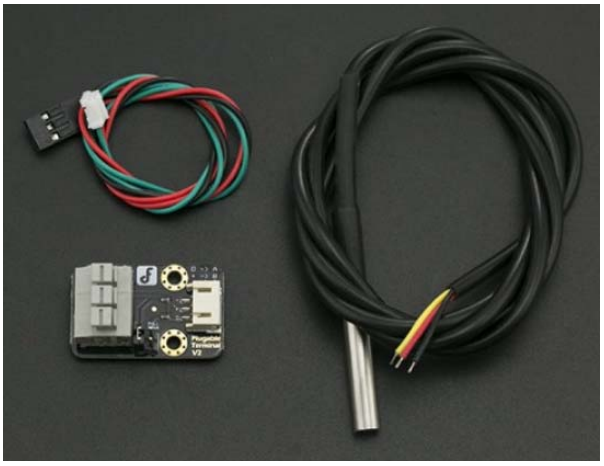
Se ajusta la placa Arduino UNO R3 para la recepción de datos a través del puerto analógico A2, activando a su vez el sensor SEN0189. Al detectar la señal analógica, comienza a registrar la variable de turbidez y manda dichos datos obtenidos por su pin analógico a la placa Arduino UNO R3. Si no es encontrada la señal, no se registran, ni transmite los datos. El programa se ejecuta de manera cíclica.



3.6.6 Esquema Placa Sensor.

3.7 SISTEMA DE DETECCIÓN DE TEMPERATURA.

Sensor de Temperatura. DS18B20



3.7.1 Descripción.

El sensor de temperatura es un dispositivo que permite conocer el valor de temperatura presente en un ambiente acuático, a través de la conversión de los cambios de temperatura a señales eléctrica, esta información es procesada por dispositivos electrónicos según la necesidad, como es el caso del Arduino UNO R3.

Es un dispositivo creado por la compañía Maxim Integrated, que tiene la capacidad de comunicarse a través de señal digital hacia el elemento electrónico que necesita la obtención de la temperatura. Cuenta con tres terminales: Vcc, GND y el pin Data.

El sensor utiliza la comunicación OneWire para envío de datos a través de un solo hilo, a diferencia de otros dispositivos que utilizan protocolos de comunicación de dos vías (Rx/Tx).

El sensor puede ser conectado a cualquier dispositivo microcontrolador tal como Arduino directamente, sin embargo la comunicación entre estos es a través del protocolo One Wire (1-wire) diseñado por Dallas semiconductor, en nuestro caso, el sensor de temperatura se va a conectar a la placa de Arduino mediante una placa conversor suministrada junto con el medidor de conductividad que utiliza la temperatura.

El sensor utilizado es una versión impermeabilizada del sensor de temperatura DS18B20. El sensor resiste hasta 125 ° C y el cable está encamisado en PVC por lo que sugerimos mantenerla por debajo de 100 ° C.
Debido a que son digitales, no se recibe ninguna señal de degradación, incluso a través de largas distancias.

El DS18B20 Proporciona 9 a 12 bits lecturas de temperatura (configurable) a través de un interfaz 1-Wire, por lo que las necesidades son solo un conductor (y tierra) para ser conectado a un microprocesador central a la tensión de 3.0-5.5V.

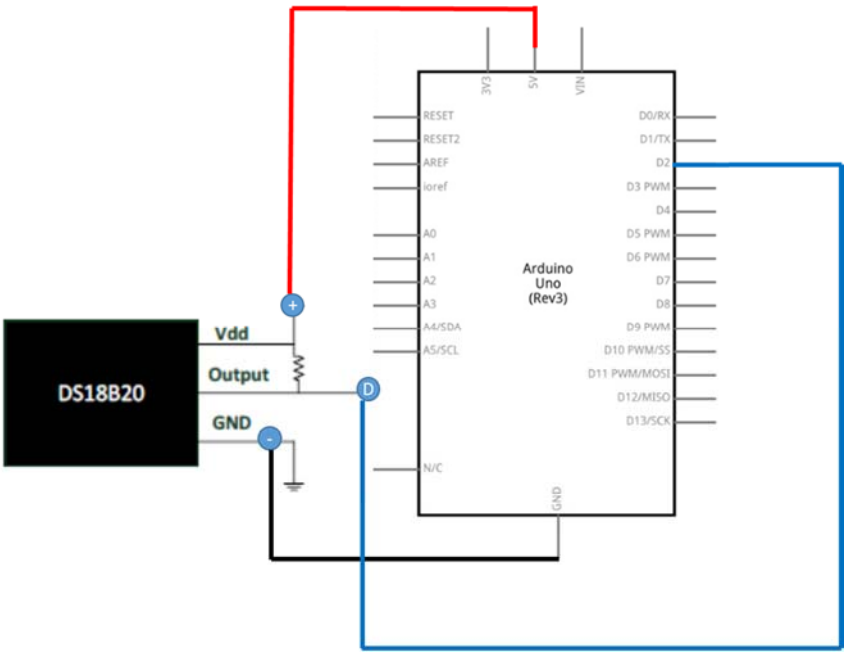
3.7.2 Características.

VOLTAJE DE FUNCIONAMIENTO:	3.0 ~ 5.5V.
PRECISIÓN	± 0,5 ° C de -10 ° C a + 85 ° C
UTILIZABLE RANGO DE TEMPERATURA	-55 a 125 ° C (-67 ° F a + 257 ° F)
RESOLUCIÓN SELECCIONABLE	9 a 12 bits
UTILIZA 1-WIRE INTERFAZ	Requiere sólo un pasador para la comunicación digital
SENSORES MÚLTIPLES	Pueden compartir una clavija
SISTEMA DE ALARMA-LÍMITE DE TEMPERATURA	
TIEMPO DE CONSULTA	Menos de 750 ms
CABLES	3 cables de interfaz
	Cable rojo – VCC Cable amarillo – GND El cable verde - DATOS
DIÁMETRO DEL CABLE	4 mm (0,16 ")
INOXIDABLE DE DIÁMETRO DE TUBO DE ACERO	de 6 mm por 35 mm (1,34 ") de largo
LONGITUD	90 cm (35.43 ")

3.7.3 Diagrama de bloque

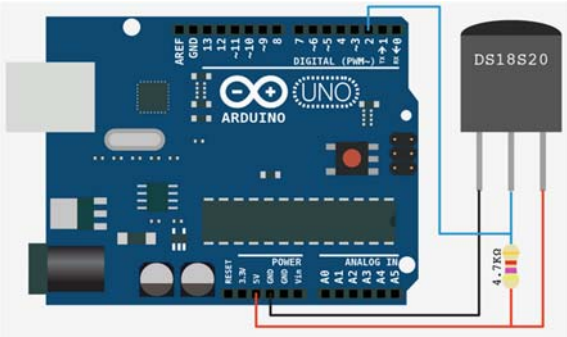
Para lograr el objetivo de monitorear en tiempo real la temperatura, en la figura siguiente se presenta el diagrama a bloques propuesto de acoplamiento entre el sensor **DS18B20** y la placa Arduino UNO R3 a través del pin Digital número D2 con una comunicación 1-wire.

Considerando las especificaciones del fabricante del sensor se coloca una resistencia de 4,7Kohms entre la entrada de energía y la salida de datos del sensor con el fin de mantener un correcto funcionamiento.



3.7.4 Esquema de conexionado

Para el correcto funcionamiento del sensor hay que poner una resistencia de 4.7K del pin de Datos y Vcc, Normalmente este sensor viene blindado en un cable largo para aplicaciones donde es necesario sumergirlo en líquidos u otras sustancias. Esta presentación del sensor solo trae 3 terminales o cables de conexión, El pin de Vcc es el cable Rojo, GND es el cable Negro y el Cable de datos puede ser de color Amarillo o Blanco.



3.7.5 Esquema Placa Sensor.

MONITORIZACIÓN DE LA CALIDAD DE AGUAS PLUVIALES VERTIDAS A CAUCE PÚBLICO EN ENTORNOS INDUSTRIALES MEDIANTE TECNOLOGÍA LOW COST

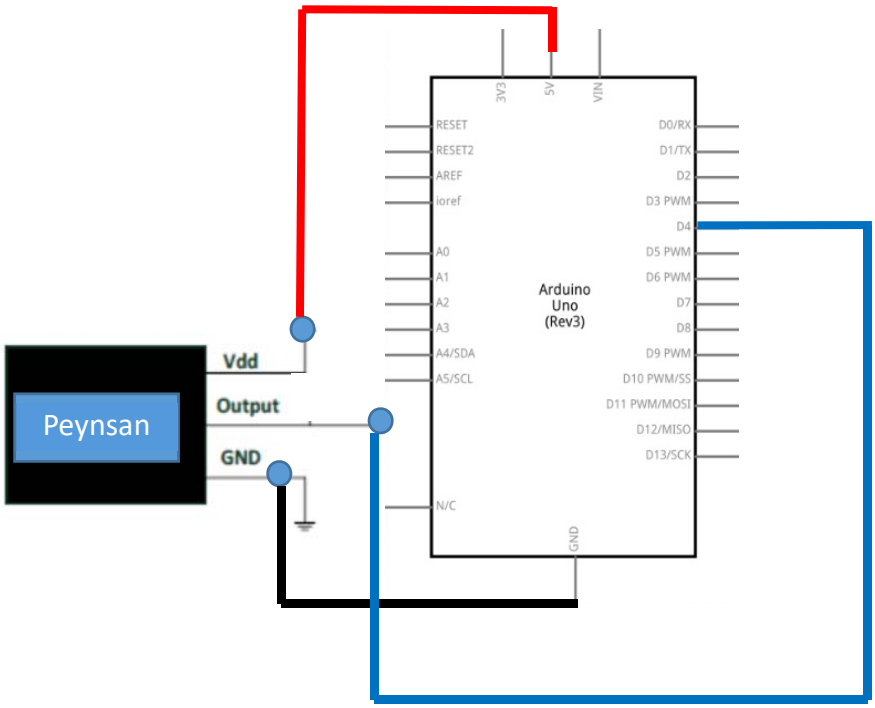
Autor: Inmaculada González

3.7.6 Pluviómetro

El instrumento utilizado para medir la precipitación se denomina pluviómetro. Un pluviómetro es un aparato que está formado por una especie de vaso en forma de embudo profundo que envía el agua recogida a un recipiente graduado donde se va acumulando el total de la lluvia caída.

3.7.7 Diagrama de bloque

Para lograr el objetivo de monitorear en tiempo real la precipitación, en la figura siguiente se presenta el diagrama a bloques propuesto de acoplamiento entre el sensor **Pynsanet** y la placa Arduino UNO R3 a través del pin Digital número D3 con una comunicación 1-wire.



Autor: Inmaculada González

4 CAPITULO 3: TECNOLOGÍA ARDUINO.

4.1 INTRODUCCIÓN.

Los dos sistemas de mayor relevancia en la actualidad de microcontroladores de bajo coste podemos decir que son las Raspberry Pi y los Arduinos.

Entre estos dos sistemas considero que el sistema Arduino es sin duda una perfecta elección para nuestro prototipo porque está más orientado al uso de adquisición de datos de forma digital y analógica que una Raspberry, que tiene un mayor parecido a lo que puede ser un ordenador.

Por otro lado al tener el Arduino una amplia comunidad y fácil acceso a la plataforma la hacen ser la mejor opción para este proyecto. Esto no quiere decir que no se puedan hacer proyectos de calidad, ni mucho menos, solo hace falta darse una vuelta por la web para poder comprobar como con estas pequeñas placas se logran proyectos de una alta calidad.

Otra característica fundamental a la hora de adquirir una placa Arduino es su bajo coste y su puesta en funcionamiento. En el mercado encontramos kits de iniciación con una gran documentación que facilita al usuario principiante, iniciarse de una forma fácil en este mundo. Esto es clave porque el usuario obtiene resultados prácticamente nada más conectar la placa, lo que provoca que se siga investigando a la vez que se van adquiriendo grandes conocimientos, por lo que la curva de aprendizaje es claramente al alza.

Es ideal también para realizar prototipos en el mundo del IoT la gran mayoría de las plataformas IoT traen compatibilidad con estas placas hardware por lo que es un candidato muy serio a la hora de aventurarnos en el internet de las cosas.

Es por ello que se ha elegido para nuestra aplicación, siendo un candidato perfecto para demostrar mediante un prototipo los usos y posibilidades que tienen dentro del Internet de las cosas.

A continuación se describe el diseño del Hardware y el desarrollo del software utilizado para la monitorización de los sensores de aguas pluviales, abarcando desde el código de control de la maqueta hasta el código empleado en la aplicación móvil. El capítulo se estructura en tres puntos principales:

Desarrollo del código Arduino. Se estudian los diferentes bloques de código que componen el sketch del microcontrolador, y se explican las clases implementadas para el control del recinto y la comunicación con el equipo servidor.

Configuración del servidor ThingSpeak. Se aborda la configuración de la plataforma ThingSpeak para poder adquirir los datos de los sensores del microprocesador *Arduino*, la comunicación con los terminales móviles, y la conexión y acceso de la aplicación servidor a la base de datos *MySQL*.

Desarrollo de la **aplicación Android**. Se analizan las diferentes *activities* que conforman la aplicación móvil, estudiando los *layouts* que las integran y su código *java* fundamental asociado.

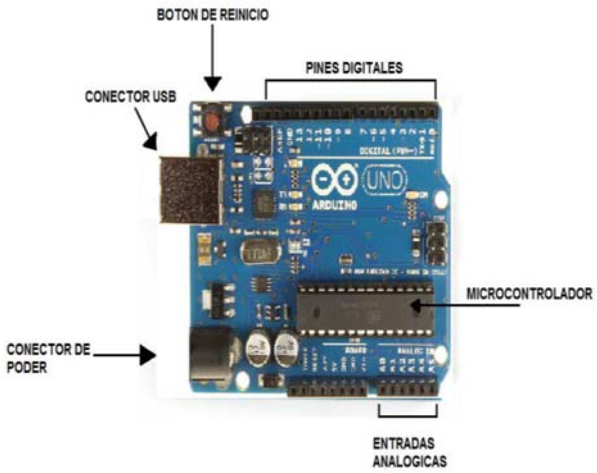
4.2 ARDUINO: HARDWARE.

4.2.1 Concepto Arduino.

Arduino es una plataforma de hardware libre, sin duda la más popular del mercado con una amplia comunidad que hace de esta solución la más apropiada para iniciarse en el mundo del hardware libre.

Se basa en una placa con un microcontrolador, con entradas y salidas analógicas y digitales, además posee un entorno de desarrollo propio, con el fin de facilitar el desarrollo de proyectos.

En la figura podemos apreciar una placa Arduino Uno, se señala algunas de las principales partes de esta placa. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). El software incluye librerías de acceso a la placa.



La placa de Arduino se caracteriza por el uso de un microcontrolador Atmel AVR, siendo el Atmel 328 uno de los más utilizados. Este modelo es el que se ha utilizado para el desarrollo del prototipo del presente TFM.

El Arduino admite los sistemas operativos más populares que hay en la actualidad, tales como Windows, Mac Os X y Linux. Sin duda una gran ventaja ya que a la hora de elegir una placa hardware, el sistema operativo no va a ser una limitación.

Su funcionamiento básico consiste en recibir información de un entorno a través de sus entradas y realizar una acción por medio de sus pines de salida. Por ejemplo se puede tener conectado un sensor de conductividad a uno de sus pines de entrada al pasar una consigna determina puede activar una alarma.

4.2.2 Hardware Arduino.

En el aspecto *hardware,* Arduino consiste en una placa con un microcontrolador y puertos de entrada/salida. Además, los módulos y accesorios proporcionan a los usuarios opciones de mejora y expansión de las placas electrónicas básicas para poder actualizarlas y readaptarlas con facilidad a nuevos objetivos. En esta sección se presentan las opciones disponibles que en la actualidad se describen en el sitio web oficial de Arduino.

4.2.2.1 Placas E/S

Arduino UNO

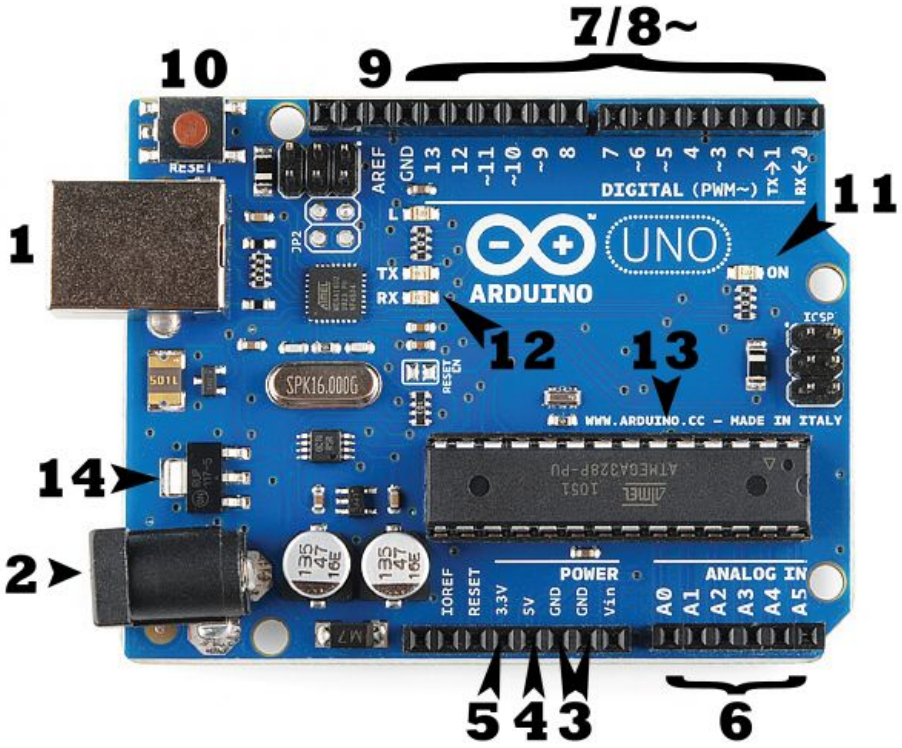
La placa que utilizaremos en nuestro proyecto es el Arduino Uno R3. Es la última revisión de la placa básica Arduino. Se conecta al ordenador por un cable USB estándar y contiene todo lo que se puede necesitar para programar y usar la placa. Pueden utilizarse multitud de módulos como extensión, que serán placas hijas con características especiales.

Las especificaciones han sido descargadas de la página oficial de Arduino, en la siguiente tabla se muestran a continuación dichas especificaciones.

DESCRIPCIÓN	VALOR
Micro controlador	ATmega328
Voltaje de funcionamiento	5V
Voltaje de entrada recomendado	7-12V
Voltaje de entrada limite	6-20V
Pines E/S digitales	14 (6 como salida PWM)
Pines de entrada analógicos	6
Intensidad por pin	40 mA
Intensidad en pin	3.3V 50 mA
Memoria Flash	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad de Reloj	16 MHz

4.2.3 Partes del Arduino.

El Arduino uno R3, constan de un pin de recepción (RX) y uno de transmisión (TX) de datos seriales TTL (Lógica de Transistor a Transistor), así como el soporte a la comunicación de tipo SPI11 y de tipo I2C 12. Cuenta con pines para la comunicación SPI y mayor protección en su circuito.



Potencia - USB (1) / Conector de Adaptador (2).

Cada placa Arduino necesita una forma de estar alimentado eléctricamente. Esta puede ser alimentada desde un cable USB que viene de su ordenador o un cable de corriente eléctrica con su respectivo adaptador. La conexión USB es también cómo va a cargar código en su placa Arduino.

NO se debe utilizar una fuente de alimentación superior a 20 voltios, ya que se puede dañar la placa Arduino. La tensión recomendada para la mayoría de los modelos de Arduino es de entre 6 y 12 voltios.

Pines (5V, 3.3V, GND, Analog, Digital, PWM, AREF)

Los pines en la placa Arduino es donde se conectan los cables de un circuito. El Arduino tiene varios tipos diferentes de entradas, cada uno de las cuales está marcado en el tablero y utilizan para diferentes funciones:

GND (3).

Abreviatura de "tierra" (en Ingles). Hay varios pines GND en el Arduino, cualquiera de los cuales pueden ser utilizados para conectar a tierra el circuito.

5V (4) y 3.3V (5).

Son los suministros pin 5V 5 voltios de energía, y los suministros de pin 3.3V 3.3 voltios de potencia.

Analógico (6)

El área de pines en el marco del 'analógica' etiqueta (A0 a A5) es analógicas. Estos pines pueden leer la señal de un sensor analógico (como un sensor de temperatura) y convertirlo en un valor digital que podemos leer.

Digital (7)

Son los pines digitales (del 0 al 13). Estos pines se pueden utilizar tanto para la entrada digital (como decir, si se oprime un botón) y salida digital (como encender un LED).

PWM (8).

Usted puede haber notado la tilde (~) al lado de algunos de los pines digitales (3, 5, 6, 9, 10 y 11). Estos pines actúan como pines digitales normales, pero también se pueden usar para algo llamado Modulación por ancho de pulsos (PWM, por sus siglas en Ingles).

AREF (9).

Soportes de referencia analógica. La mayoría de las veces se puede dejar este pin solo. A veces se utiliza para establecer una tensión de referencia externa (entre 0 y 5 voltios) como el límite superior para los pines de entrada analógica.

Botón de reinicio (10)

Empujando este botón se conectará temporalmente el pin de reset a tierra y reinicie cualquier código que se carga en el Arduino. Esto puede ser muy útil si el código no se repite, pero quiere probarlo varias veces.

Indicador LED de alimentación (11)

Este LED debe encenderse cada vez que conecte la placa Arduino a una toma eléctrica. Si esta luz no se enciende, hay una buena probabilidad de que algo anda mal.

Leds RX TX (12)

TX es la abreviatura de transmisión, RX es la abreviatura de recibir. Estas marcas aparecen un poco en la electrónica para indicar los pasadores responsables de la comunicación en serie. En nuestro caso, hay dos lugares en la Arduino UNO donde aparecen TX y RX - una vez por pines digitales 0 y 1, y por segunda vez junto a los indicadores LED de TX y RX (12). Estos Leds nos darán algunas buenas indicaciones visuales siempre nuestro Arduino está recibiendo o transmitiendo datos (como cuando nos estamos cargando un nuevo programa en el tablero).

Microcontrolador (13)

Lo negro con todas las patas de metal es un circuito integrado (IC, por sus siglas en Ingles). Piense en ello como el cerebro de nuestro Arduino. La principal IC en el Arduino es ligeramente diferente del tipo de placa a placa tipo, pero es por lo general de la línea de ATmega de CI de la empresa ATMEL. Esto puede ser importante, ya que puede necesitar para saber el tipo de IC (junto con su tipo de tarjeta) antes de cargar un nuevo programa desde el software de Arduino. Esta información se puede encontrar en la escritura en la parte superior de la IC. Si quieres saber más acerca de la diferencia entre diversos circuitos integrados, la lectura de las hojas de datos suele ser una buena idea.

Regulador de Voltaje (14)

Esto no es realmente algo que se puede (o debe) interactuar con el Arduino. Pero es potencialmente útil para saber que está ahí y para qué sirve. El regulador de voltaje hace exactamente lo que dice - que controla la cantidad de tensión que se deja en la placa Arduino. Piense en ello como una especie de guardián; se dará la espalda a una tensión adicional que podría dañar el circuito. Por supuesto, tiene sus límites, por lo que no conecta tu Arduino a nada superior a 20 voltios.

4.2.3.1 Módulos.

***Arduino* GPRS/GSM Shield**

Conecta la placa Arduino a Internet utilizando la red inalámbrica GPRS. Sólo hay que conectar este módulo a la placa Arduino, conectar una tarjeta SIM de un operador que ofrezca cobertura GPRS y seguir unas sencillas instrucciones para empezar a controlarlo a través de internet. También puede realizar/recibir llamadas de voz (se necesita un circuito de altavoz y micrófono externo) y enviar/recibir mensajes SMS.

En la figura podemos apreciar el modelo del GPRS/GSM SIM900 que utilizaremos para nuestro proyecto. Este módulo nos permitirá enviar y recibir mensajes con ciertas instrucciones para nuestro prototipo de alarma basada en Arduino. Estos escudos interactúan con la placa de Arduino permitiendo de esta manera manejar las mismas salidas y entradas que tiene la placa Arduino.

4.3 ARDUINO: SOFTWARE.

4.3.1 Código abierto.

Al profundizar en el mundo Arduino podemos afirmar que esta tecnología engloba tres herramientas específicas.

1- La primera es el controlador de Arduino que existe en varios tamaños desde pequeños hasta grandes. Cuenta con su esquema de libre acceso. Cualquier persona con los conocimientos necesarios podría ensamblar este controlador.
2- En segundo lugar tenemos el lenguaje y el compilador que crea código para el controlador que simplifica muchas de las tareas que se presentan como un desafío para los diseñadores y programadores cuando trabajan con un hardware e interacción física.
3- Y por último tenemos el entorno de programación (IDE), es de código abierto desarrollado en Java.

4.3.2 Multiplataforma.

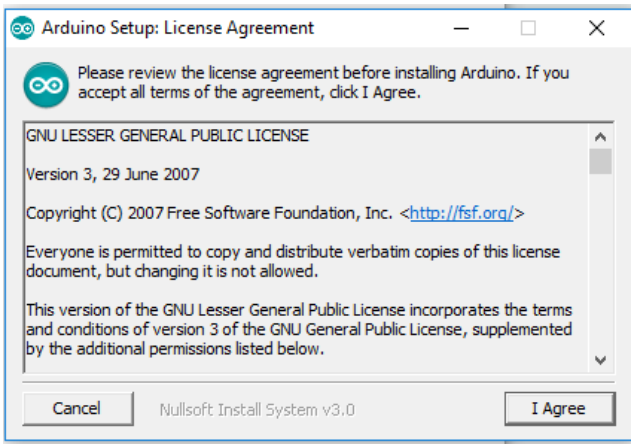
Arduino utiliza librerías de código abierto, escritas en C y C++ y compiladas para diferentes sistemas operativos. Además el entorno de desarrollo está escrito en Java lo que permite tener un software multiplataforma.

El IDE de Arduino puede ser instalado en los sistemas operativos Windows, en MAC y Linux

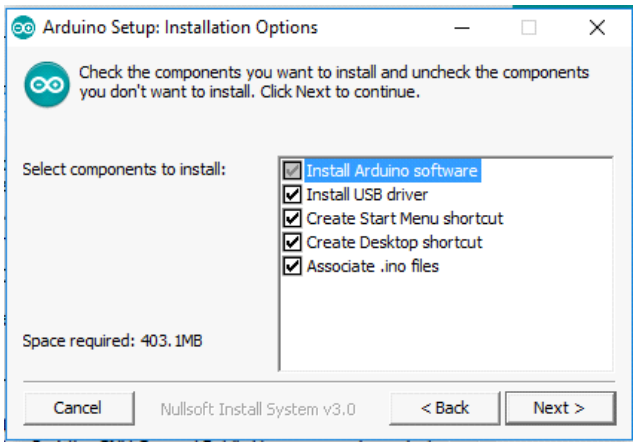
4.3.3 Entorno

Para instalar Arduino descargamos el instalador de la página web oficial de Arduino https://www.arduino.cc/en/Main/Software#.

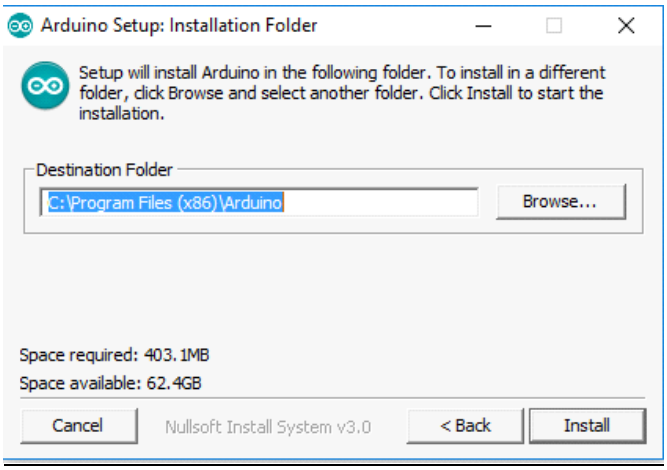
Se nos descargara el archivo ejecutable arduino-1.0.6-windows.exe. Como pre requisito debemos tener instalado el JDK de Java en nuestra ordenador, las pantallas de instalación son sencillas como podremos apreciar en siguientes figuras, finalmente en la última figura se muestra el entorno de desarrollo de Arduino.



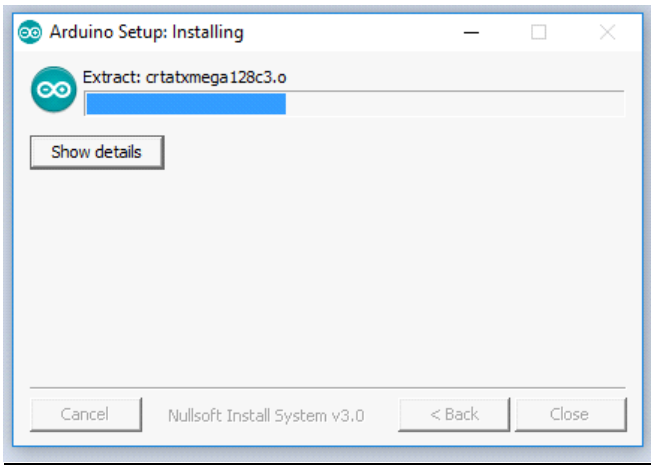
Proceso de instalación de IDE para Arduino



Proceso de instalación de IDE para Arduino

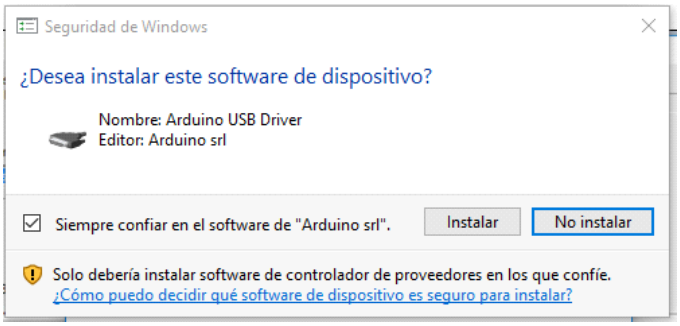


Proceso de instalación de IDE para Arduino

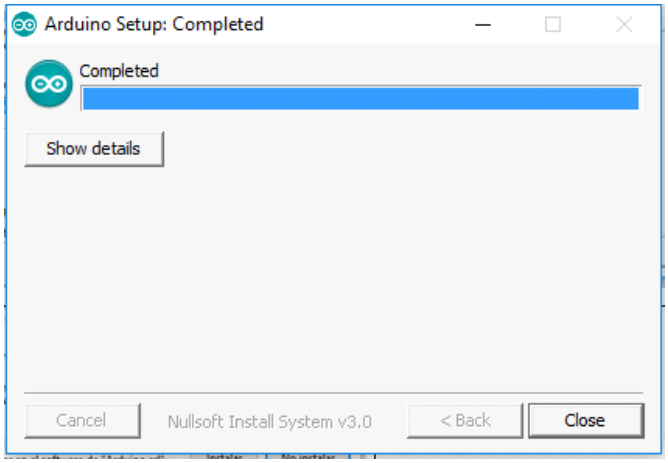


Proceso de instalación de IDE para Arduino

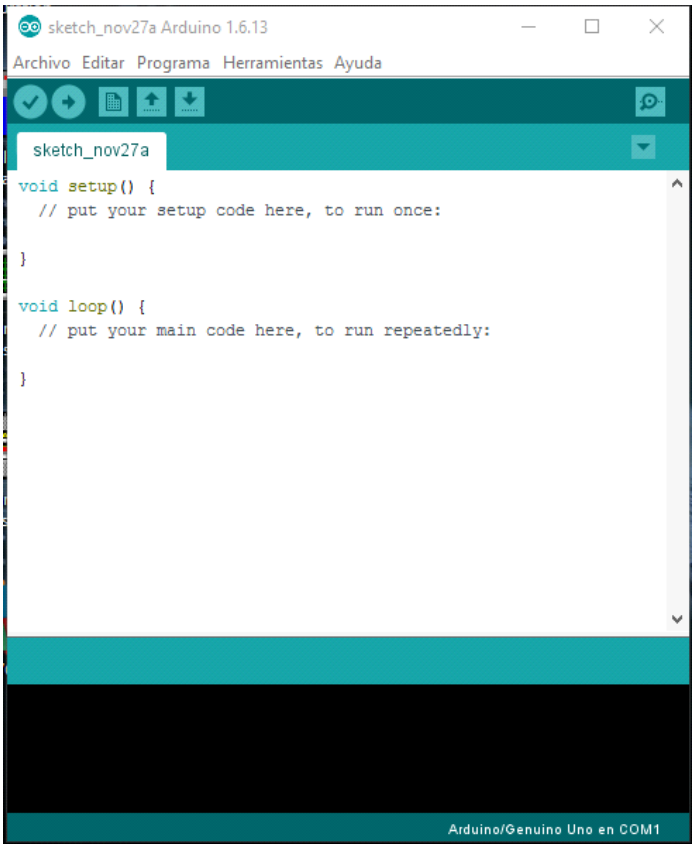
Autor: Inmaculada González



Proceso de instalación de IDE para Arduino



Proceso de instalación de IDE para Arduino



IDE de Arduino ejecutado

4.4 DESARROLLO DEL CÓDIGO ARDUINO.

Para funcionar, las placas Arduino necesitan que se carguen los *sketches* en sus microcontroladores, y para ello es necesario el uso del *software* de Arduino, que se encuentra disponible de forma gratuita en su página web oficial. Si la placa necesita una conexión USB con el ordenador, se deberán instalar los drivers USB para el chip FTDI de la placa, que también pueden encontrarse en la página de Arduino.

Un programa diseñado para ejecutarse sobre una placa *Arduino* (un "*sketch*") siempre se compone, al menos, de tres secciones:

1) La sección de declaraciones de **variables globales**: ubicada directamente al principio del sketch.
2) La sección llamada **"void setup()"**: delimitada por llaves de apertura y cierre.
3) La sección llamada **"void loop()"**: delimitada por llaves de apertura y cierre.

4.4.1 Estructura.

4.4.1.1 Librerías.

Inserción de ficheros externos que contengan funciones que vayan a utilizarse en el código principal.

Por ejemplo, para leer el sensor DS18B20 con un Arduino es necesario utilizar dos librerías que deben ser instaladas antes de cargar el código a nuestra placa de desarrollo. Las librerías son las siguientes:

- Dallas Temperatura.
- OneWire

4.4.1.2 #define.

Permite establecer variables constantes de forma que, al compilar el fichero el microprocesador, se sustituyan las referencias a la variable *#define* por el valor que tengan asociado.

4.4.1.3 Variables globales.

Encierra variables de ámbito global, es decir, variables que tienen visibilidad desde cualquier parte del programa. Son, por lo general, variables que hacen referencia a **estados del sistema o situaciones** que afectan al funcionamiento general del programa.

4.4.1.4 Definición de clases.

Este bloque encierra las implementaciones de las clases que van a usarse en el código. Tras su declaración, se pueden crear instancias de estas (*objetos*) en el programa principal para su utilización, incluso pueden crearse instancias dentro de otras clases, aumentando las posibilidades y opciones de estas.

4.4.1.5 Creación de objetos.

En esta sección del código se crean las instancias de las clases, y por tanto se definen los *objetos globales* que se utilizarán en el programa.

4.4.1.6 Declaración Setup().

Se ejecuta una sola vez al conectar o resetear la placa.

4.4.1.7 Declaracion loop().

Contiene todo lo que se necesita hacer en repetidas ocasiones dentro de la aplicación, como por ejemplo chequear un nuevo valor desde un control, enviar información a la computadora, enviar una señal a un pin.

Por tanto, las instrucciones escritas en la sección "*void setup()*" normalmente sirven para realizar ciertas preconfiguraciones iniciales y las instrucciones del interior de "*void loop()*" son, de hecho, el programa en sí que está funcionando continuamente.

Además de estas funciones básicas, pueden implementarse nuevas **funciones**, de forma que se puede encapsular tareas específicas que se necesiten utilizar de forma periódica en la función *loop ()*

Estas funciones se pueden crear directamente en el fichero principal, o pueden utilizarse **librerías** ya implementadas, las cuales ofrecen funciones ya programadas destinadas a tareas concretas (p.ej lectura de sensores, conversión de variables…).

4.4.1.8 *Definición de interrupciones.*

En la parte final del código se definen las funciones de tratamiento de interrupciones. Inmediatamente después de detectar una interrupción en el pin configurado, se invoca la función asociada al número de la interrupción11 y una vez que se ejecuta su contenido, el programa retorna al punto en el que se lanzó la interrupción.

4.4.1.9 Constantes.

Arduino tiene las siguientes constantes que pueden ser utilizadas en cualquier parte del código de una aplicación:

HIGH/LOW.- Estas definen el nivel de voltaje en un pin digital, pudiendo ser 5V o 0V.

INPUT/OUTPUT.- Estas son constantes que son usadas para definir un pin, entrada o salida.

4.4.1.10 Métodos

pinMode(número_de_pin, mode).

Los pines digitales del controlador Arduino pueden ser definidos como entrada (INPUT) o salida (OUTPUT). Debemos establecer si será de entrada o de salida en el setup (). Generalmente solo lo hacemos una vez en la aplicación. Por ejemplo: pinMode (11, OUTPUT) en este ejemplo el pin 11 está puesto como salida, todo esto debemos declararlo dentro de setup ().

digitalWrite(valor, pin).

Este método coloca un pin digital en HIGH o en LOW dependiendo del uso que le daremos. Para poder ser colocados estos valores primero deben haber sido definidos en el método pinMode (). Por ejemplo:.void setup().{pinMode(2, OUTPUT);digitalWrite(2, LOW);}.

int digitalRead(numero_de_pin)

Este es un método que lee el estado de un pin que está en modo INPUT. Retorna el valor del pin como un integer (valor entero).

analogRead(numero_de_pin)

Este método lee el valor de un pin analógico. Un pin analógico puede tener un valor desde 0 a 1023 que representa el voltaje del pin. El valor retornado será un integer. Por ejemplo podemos declarar un int y recibir lo que devuelva el método. int pin_11=analogRead(11);

4.4.2 Sintaxis

La página oficial de Arduino nos da como referencia de su sintaxis unos cuatro ejemplos.

- **;(punto y coma).-** Utilizado para terminar una declaración.
- **{} (llaves).-** Una llave de apertura "{" siempre debe ir seguida de una llave de cierre "}". Esta es una condición a la que se suele referir como llaves emparejadas. El IDE (Entorno Integrado de Desarrollo) Arduino incluye una característica para comprobar si las llaves están emparejadas. Sólo tienes que seleccionar una Llave o incluso hacer click en el punto de inserción que sigue inmediatamente a una llave, y su compañera lógica será seleccionada. En la actualidad esta característica tiene un pequeño fallo, el IDE encuentra a menudo (incorrectamente), llaves en el texto que pueden estar situadas dentro de comentarios.
- **// (comentarios en una línea).-** Los comentarios son líneas en el programa para aclarar sobre el funcionamiento del programa. Estas líneas son ignoradas por el compilador y no se exportan al procesador.
- **/**/ (comentarios en múltiples líneas).-** Los comentarios son líneas en el programa para aclaraciones sobre el funcionamiento del programa. Estas líneas son ignoradas por el compilador y no se exportan al procesador. Para resumir a continuación la tabla 4.1 nos muestra operadores aritméticos, operadores comparativos, tipos de datos con los que se puede trabajar en el IDE de Arduino.

4.4.2.1 Sintaxis de lenguaje de programación de Arduino.

Tipos de datos	Operadores Aritméticos	Operadores comparativos
boolean (booleano)	= (asignación)	== (igual a)
char (carácter)	+ (suma)	!= (distinto de)
byte	- (resta)	< (menor que)
int (entero)	* (multiplicación)	> (mayor que)
unsigned int (entero sin signo)	/ (división)	<= (menor o igual que)
long (entero 32b)	% (resto)	>= (mayor o igual que)
unsigned long (entero 32b sin signo)		
float (en coma flotante)		
double (en coma flotante de 32b)		
string (cadena de caracteres)		
array (cadena)		
void (vacío)		

4.5 COMUNICACIONES: PROTOCOLO GRPS.

Como medio para el envío y recepción de datos a la plataforma ThingSpeak utilizaremos el protocolo GRPS.

El Protocolo GPRS significa General Packet Radio Service o en español Servicio General de Paquetes por Radio. Orientado a la transmisión de datos. Los proveedores de este servicio cobran por la cantidad de datos enviados o recibidos más no por un enlace permanente. Dicho en otras palabras un usuario utiliza un canal solo cuando va a transmitir datos. Luego de la rápida expansión de las redes GSM que utilizan la

Conmutación de circuitos para el servicio de llamadas de voz, los usuarios necesitaron servicios más avanzados como la transmisión de datos.

4.5.1 Red GPRS

El sistema GPRS nació para darle al usuario la oportunidad de poder transmitir datos.

GPRS comparte el mismo rango de frecuencias que GSM. Repasaremos los componentes principales de una red GSM:

- **MS (Mobile Station).-**Equipo físico usado por el usuario.
- **SIM (Subscriber Identity Module).-** Es el chip identificador del abonado. Contiene información referente solo al usuario.
- **BTS (Base Transceiver Station).-** Establece vía radio conectividad entre las estaciones móviles y la red GSM.
- **BSC (Base Station Controller).**- Actúa como un concentrador de varias estaciones base (BTS).
- **BSS (Base Station Subsystem).**- Está constituido por la BTS y la BSC.
- **MSC (Mobile Switching Center).-** Centro de Conmutación de servicios móviles. Control de llamadas y encargado de la asignación de canales de usuario entre el MSC y el BSC.
- **GMSC (Gateway MSC).-** Es a través del cual se conecta la red GSM con las redes fijas.
- **HLR (Home Location Register).**- base de datos que contiene los datos de los clientes para su gestión. Contiene información como: servicios contratados, limitaciones de servicio y localización del usuario.
- **VLR (Visitor Location Register).-** Contiene datos dinámicos que permiten saber la última ubicación del usuario.

La red GPRS agrega algunos componentes a la red GSM tales como el SGSN (responsable de la transferencia de paquetes) y el GGSN (convierte los paquetes al formato IP). La inclusión de estos nuevos componentes permite la conmutación de paquetes y la utilización de protocolos como IP.

5 CAPITULO 4: PLATAFORMA GRATUITA PARA EL REGISTRO DE LOS VALORES DE SENSORES.

5.1 INTRODUCCIÓN.

La necesidad de tener registrados los datos y sus acceso online desde cualquier equipo que tenga conexión a internet lleva a plantear la busca de una plataforma gratuita que nos permita tener los parámetros registrados y monitorizados.

Dentro de la búsqueda de las plataformas actuales existe en la actualidad nueva revolución tecnológica, dónde objetos cotidianos del día a día con una función específica, evolucionan gracias al IoT, pasando a estar conectados y dotar de nuevas funcionalidades a estos objetos, pudiéndolos controlar y administrar desde tabletas, ordenadores o teléfonos móviles.

La plataforma IoT (Internet of Things) "es un concepto que abarca un conjunto de tecnologías que buscan permitir la interconexión e interoperabilidad de objetos heterogéneos y ubicuos a través de diferentes redes."

Con la plataforma IoT cobran también mucha importancia los Smart Objects: objetos físicos con un sistema embebido que le permite procesar información y comunicarse con otros dispositivos y realizar acciones en base a una acción o evento determinado. Estos pueden ser los Smartphones, un micro-controlador como es Arduino o cualquier otro objeto que tenga conectividad a la red (Lee & Kim, 2012) y sea capaz, como mínimo, de gestionar información (Meyer, Främling, & Holmström, 2009). Pues, en combinación con Internet of Things, se consigue que la red pase de solo transportar datos, a dotarla de inteligencia y acciones según los datos que recogen estos objetos y los servicios que estos permiten realizar.

Con el uso de la plataforma IoT se pretende como objetivo final tener objetos que nos den información desde cualquier lugar del mundo. Con esto se puede ofrecer la interconexión de estos objetos heterogéneos a través de una red de comunicación, como puede ser Internet (Lee & Kim, 2012). Estos objetos pueden ser sensores dispersos por el terreno, una red de sensores, un micro-controlador Arduino, un Smartphone, ya que Internet of Things es la interconexión de objetos heterogéneos a través de Internet.

Como ejemplo del uso de esta tecnología está el presente trabajo fin de master, conectando sensores para el monitoreo ambiental.

Objetivo del uso de plataformas IoT en el proyecto es desarrollar una aplicación para visualizar y registrar los parámetros medidos en los tanques de aguas pluviales, a través de un microcontrolador conectándonos a una plataforma software. Otro de los puntos interesantes que se mostraran con la aplicación, será el tratamiento de los datos IoT.

Este podría ser crucial para la prevención o control de cierto tipo de circunstancias en las que podrían jugar un papel crucial (Broring, Maue, Malewski, & Janowicz, 2012). Por ejemplo, conociendo en cada momento los valores de aquellos parámetros relevantes, se podría evitar que un agua pluvial contaminada por el entorno industrial se vertiese a una ría catalogada protegida en la RED NATURA 2000, analizando la calidad del agua y bombeando la misma hasta una PTA para evitar problemas mayores
No obstante, todos estos sistemas son muy complejos. El principal problema a la hora de interconectar los Smart Objects para crear una red inteligente es el trato de los diferentes valores y tipos de mensajes que envía cada dispositivo. Esto es debido tanto al diferente hardware utilizado por cada uno como al software que utiliza. Esto se debe a las normas en su creación por parte de los diferentes fabricantes (Gama, Touseau, & Donsez, 2012).
Para solventar el problema la heterogeneidad de los objetos que dificulta la comunicación entre los mismos, que presenta el uso de la tecnología Internet of Things y los Smart Objects para ser utilizados como una red inteligente que controle casi cualquier cosa. Se utiliza una de las soluciones actualmente más populares, para conectar objetos es la realización de una interfaz en común. Será una aplicación que resida en ambos y consulte un servidor.

5.2 TECNOLOGÍA IOT.

El internet de las cosas se basa en sensores, en redes de comunicaciones y en una inteligencia que maneja todo el proceso y los datos que se generan. Los sensores son los sentidos del sistema y, para que puedan ser empleados de forma masiva, deben tener bajo consumo y coste, un reducido tamaño y una gran flexibilidad para su uso en todo tipo de circunstancias

En la actualidad, se ha detectado que cada vez más Smart Objects pueden conectarse a Internet. Estos pueden interactuar entre ellos, enviarse datos y realizar determinadas acciones según cierta información. Objetos heterogéneos interactuando entre ellos: esto es Internet of Things.

Por tanto, podemos decir que el Internet de las Cosas (IoT) consiste en la integración de sensores y dispositivos en objetos cotidianos que quedan conectados a Internet a través de redes fijas en inalámbricas. Dado su tamaño y coste, los sensores son fácilmente integrables en cualquier aplicación que se nos ocurra diseñar, de esta manera, cualquier objeto es susceptible de ser conectado y estar presente en la Red.

El concepto de Internet de las cosas fue propuesto por Kevin Ashton en el Auto-ID Center del MIT en 1999, donde se realizaban investigaciones en el campo de la identificación por radiofrecuencia en red (RFID) y tecnologías de sensores. Surgió en base a la necesidad de las cadenas de suministro y la identificación de objetos, personas y animales mediante el uso de etiquetas inteligentes RFID. Esto se debe a que con el uso de RFID se puede otorgar un identificar único a un objeto. De esta manera, se puede conseguir localizar a un objeto concreto para determinadas tareas.

No obstante, para la existencia de Internet of Things, se necesitan tres pasos:

- Inteligencia integrada.
- Conectividad.
- Interacción.

Por ello, la base de Internet of Things son los sensores y las tarjetas RFID.

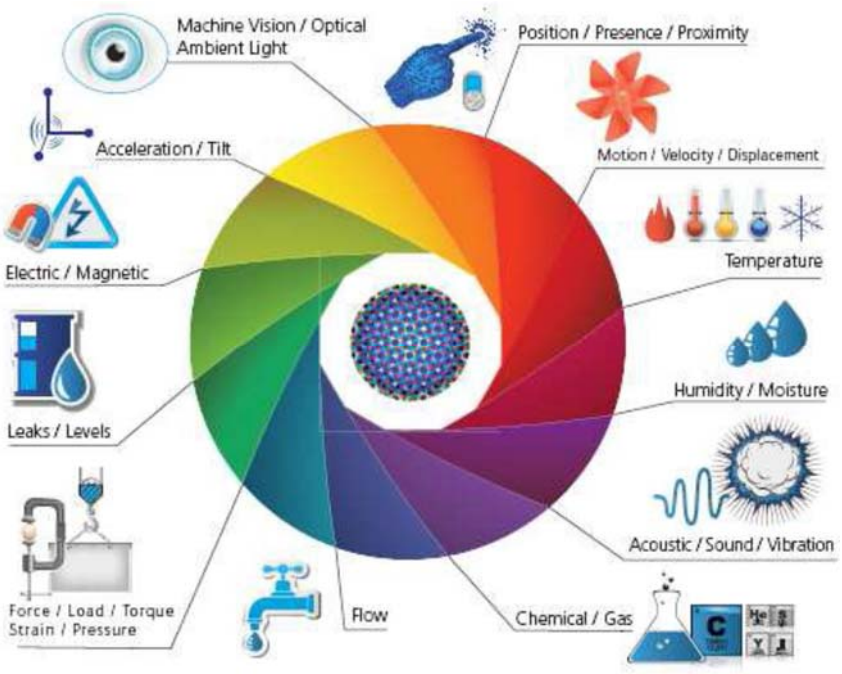
Los sensores sirven para la recogida casi de cualquier tipo de información, cambio o dato. Este puede ser tratado por un Smart Object, como puede ser un Smartphone o enviado a un servidor.

Las etiquetas inteligentes RFID se puede identificar diferentes objetos, para que, por medio de radiofrecuencia, se lean las propiedades de este elemento, e incluso, sea haga un seguimiento de él. Si combinamos ambos, se puede hacer que un objeto concreto realice una acción en base a un evento captado por un sensor determinado.

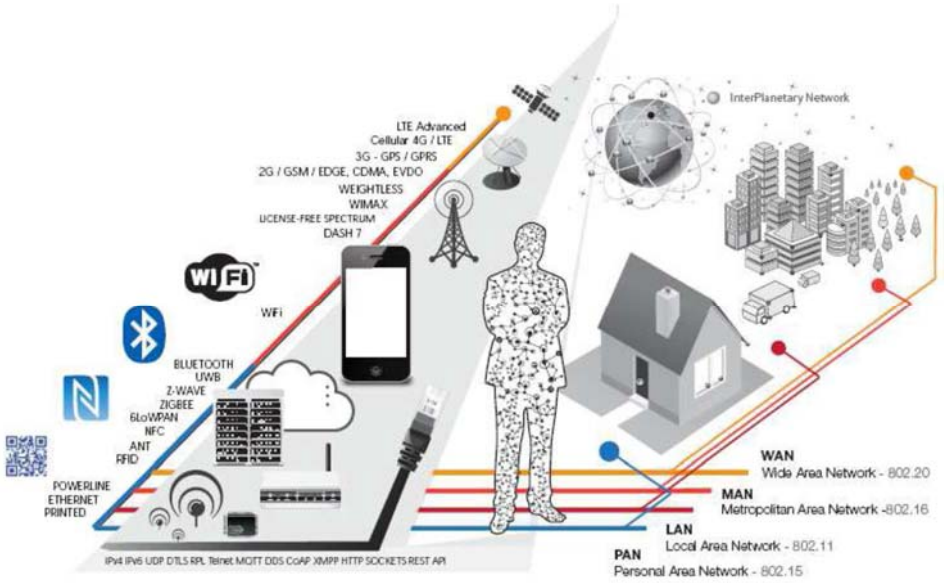
Por ello se necesita de una infraestructura inteligente que sea capaz de conectar los diferentes objetos y les dote de la inteligencia necesaria para interactuar entre ellos, incluso, en algunos casos, indicándoles la decisión que deben tomar (Tan, 2010). Esto último, en el supuesto de que los Smart Objects no tengan inteligencia propia o necesiten en un momento dado una decisión externa.

En las siguientes tres figuras, podemos resumir el IoT:

1- **Sensores y actuadores**: estamos dando a nuestro mundo un sistema nervioso digital. Ubicación de datos utilizando sensores GPS, los ojos y las orejas con cámaras y micrófonos, junto con los órganos sensoriales que pueden medir todo, desde la temperatura a cambios de presión.

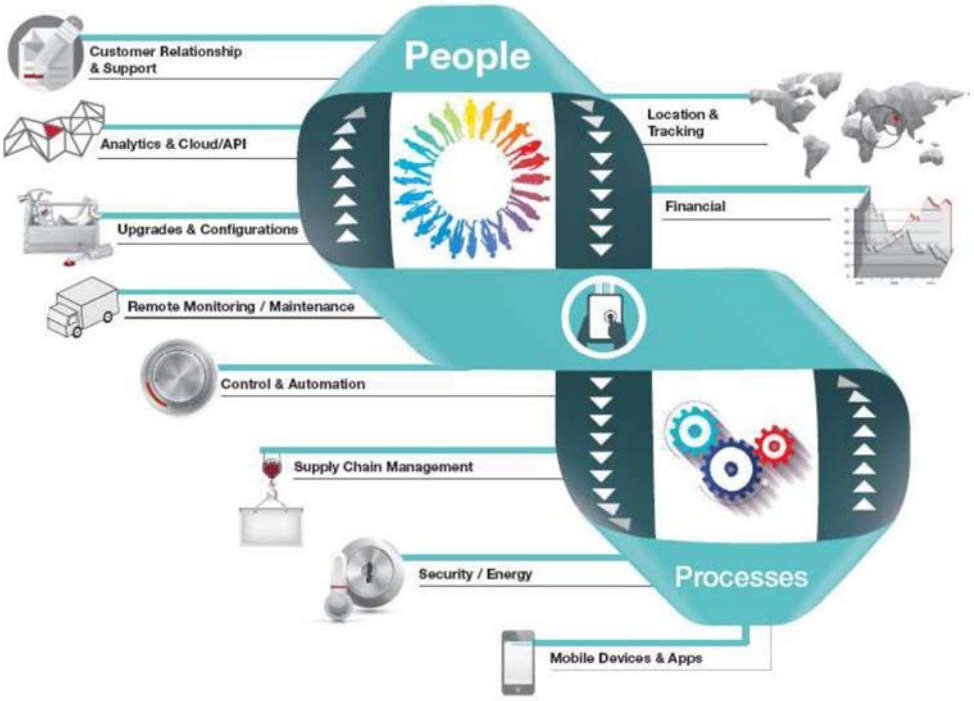


2- **Digitalización**: estas entradas se digitalizan y se colocan en las redes



3- **Personas & Procesos:** estas entradas en la red se pueden combinar en sistemas bidireccionales que integran datos, personas, procesos y sistemas para la mejora en la toma de decisiones.

MONITORIZACIÓN DE LA CALIDAD DE AGUAS PLUVIALES VERTIDAS A CAUCE PÚBLICO EN ENTORNOS INDUSTRIALES MEDIANTE TECNOLOGÍA LOW COST



Autor: Inmaculada González

5.2.1.1 Smart Objects.

Como explican numerosos autores un Smart Object, también conocido como Intelligent Product, es un elemento físico, con diversas propiedades, identificable a lo largo de su vida útil, que interactúa con el entorno y otros objetos y que puede actuar de manera inteligente según unas determinadas situaciones, mediante una conducta autónoma. Ejemplos de Smart Objects cotidianos son los Smartphones, las Tablets, Smart TVs e incluso algunos coches.

Los Smart Objects, lo clasifican en base a tres dimensiones. Cada dimensión se corresponde a un tipo de inteligente. De esta manera, por medio de las tres dimensiones de un objeto, se puede determinar la inteligencia y tipo de Smart Object que es y compararlo con otros. Las dimensiones son las siguientes:

1- La primera es **el nivel de inteligencia**. En esta se describe la capacidad de inteligencia del objeto. Consta de tres categorías: La gestión de la información: capacidad para manejar la información que recoge a través de sensores, lectores u otras técnicas; La notificación del problema, que es la posibilidad de que sea capaz de notificar a su propietario cuando ocurre un determinado problema o evento en el propio Smart Object, como puede ser la detección de bajada de temperatura; Por último, la toma de decisiones, que es cuando, además de todo lo anterior, posee la capacidad propia sobre la toma de decisiones sin intervención de un control externo.
2- La segunda dimensión es la localización de la inteligencia. Esta consta de dos categorías. La primera es la inteligencia a través de la red. Consiste en que la inteligencia del objeto depende totalmente de un agente externo al propio objeto. Este agente puede ser una red a la que se encuentra conectado. La otra categoría es la inteligencia en el objeto. Los objetos que pertenecen a esta computan todo por sí mismo, es decir, toda la inteligencia es llevada en ellos.
3- La última dimensión es la agregación del nivel de inteligencia y se compone de dos categorías: inteligencia en el elemento e inteligencia en el contenedor. A la primera pertenecen aquellos objetos que solo pueden manejar información, notificaciones y/o decisiones y si contienen otros componentes, estos no pueden ser distinguidos como objetos individuales. Por ejemplo, un sensor. Mientras, la inteligencia del contenedor, además de poder manejar información, notificaciones y/o decisiones, es capaz de seguir funcionando como elemento u objeto a pesar de que se le desensamble alguna parte de él. A este grupo pertenecería una placa Arduino con como mínimo tres sensores. Si a esta se le quita un sensor, puede seguir funcionando como contenedor.

De esta manera, se puede clasificar a un Smart Object en sus tres dimensiones. En este proyecto, se conectarán sensores con un nivel de inteligencia de gestión de la información y de notificación del

problema, con la localización de la inteligencia a través de la red y de ambos niveles de agregación del nivel de inteligencia.

Actualmente existen varias plataformas para interconectar dispositivos. Entre ellas se encuentran algunas orientadas al mundo empresarial, como es el proveedor de servicios Xively. Otras están orientadas a la investigación, como son Paraimpu y QuadraSpace. Sobre Paraimpu se puede encontrar varias publicaciones en diferentes sitios. Estas tratan los problemas principales de IoT. Por último, están las redes abiertas a los usuarios pero que están en estado beta y limitan el acceso a los usuarios mediante invitación o aprobación de la cuenta. Ejemplos de esto son ThingSpeak, Sensorpedia o SenseWeb.

5.3 PLATAFORMAS SOFTWARE: THINGSPEAK.

Para el presente trabajo se ha utilizado la red abierta ThingSpeak. Los motivos por los que se ha elegido está plataforma son:

- Por ser una plataforma Open Source.
- Por su fácil configuración. Al tener una API documentada ayuda notablemente al manejo de la misma y a su rápida configuración.
- Disponer de una integración sencilla con aplicaciones de terceros.
- Otro motivo fundamental es su compatibilidad con Arduino y el disponer de documentación referente a esta tipo de hardware.
- Y por último, que el conjunto Arduino-ThingSpeak se convierte en un sistema de Bajo Coste.

5.3.1 ThingSpeak

ThingSpeak es una plataforma abierta de aplicaciones, diseñada para permitir conectar personas con objetos. Se caracteriza por ser una plataforma Open Source con una API para almacenar y recuperar datos de los objetos usando el protocolo HTTP sobre Internet o vía LAN (Local Area Network).

Se trata de una plataforma basada en Ruby on Rails 3.0 (RoR), este es un framework de aplicaciones web de código abierto basado en Ruby, cuya arquitectura está basada en el Modelo Vista Controlador (MVC).

Se caracteriza por su simplicidad a la hora de programar aplicaciones del mundo real, escribiendo menos código y con una configuración mucho más sencilla que otros frameworks. Otra de las características que hacen de RoR un framework perfecto para el desarrollo de aplicaciones es que permite el uso de meta programación, haciendo que su sintaxis sea más legible y llegue a un gran número de usuarios.

La aplicación incluye todo lo necesario para poder empezar a trabajar, desde una aplicación web en la que podremos gestionar usuarios, gestionar claves de API, gestión de canales y cartografía.

La plataforma ThingSpeak almacena los datos en "Canales ThingSpeak". Los canales se representan como una interfaz web donde se publican los datos almacenados. Pueden ser configurados para ser públicos para que otras personas puedan verlos o privados, sólo se puede acceder mediante el registro en ThingSpeak.com con cuenta de usuario.

Las aplicaciones cliente están integradas en los dispositivos físicos que pueden leer y escribir a un canal ThingSpeak mediante peticiones HTTP a la API de ThingSpeak. Cada entrada o feed es etiquetada con una ID de entrada única y se almacenan con una fecha y un time stamp.

Escribir en un canal requiere una clave de escritura para asegurar que sólo las aplicaciones autorizadas pueden acceder a sus datos. Los datos numéricos pueden ser procesados como escala de tiempo, promedio, mediana, suman, y el redondeo. Los canales soportan los formatos JSON, XML, CSV. Para poder interactuar en base al objeto deseado, hay que bajar los datos en uno de estos tres formatos y procesarlo para así obtener los datos necesarios. Por el contrario, para enviar datos hay que crear una petición mediante protocolo HTTP, método POST y siguiendo la arquitectura REST.

5.3.1.1 Hardware compatible.

Arduino, Raspberry pi, Electric Imp, Freetonics, Netduino y Xbee wíreless networks

5.3.1.2 Características principales.

Destacaremos algunos puntos importantes en toda plataforma tales como su API, App (si las tiene), integración, hardware.

CHART API

Un punto importante a la hora de desarrollar cualquier proyecto es encontrar un API disponible de forma sencilla para que el desarrollador tenga los mecanismos necesarios para el desarrollo de la aplicación.

En este caso, ThingSpeak dispone de una **API** la cual está disponible en GitHub para su descarga en un servidor propio. Es totalmente abierta, por lo que también se puede modificar su código fuente original y así contribuir a la comunidad con nuevas características, un principio básico en toda plataforma Open Source.

CHANNELS.

La forma que tiene esta plataforma de almacenar y publicar los datos es a través de los "Chanels" (Canales). Su creación es muy simple y en un par de clicks y rellenando una serie de datos lo tendremos disponible sin mayor complicación. Nuevamente y según profundizas en este plataforma ves que el objetivo es simplificar el trabajo.

Se utiliza para mostrar los datos almacenados en los canales ThingSpeak en gráficos. Estos gráficos se denominan "charts" y pueden presentar datos numéricos y puede ser embebido en sitios web externos. Los gráficos pueden ser estáticos o dinámicos. Los gráficos dinámicos visualizan la variación de datos en tiempo real.

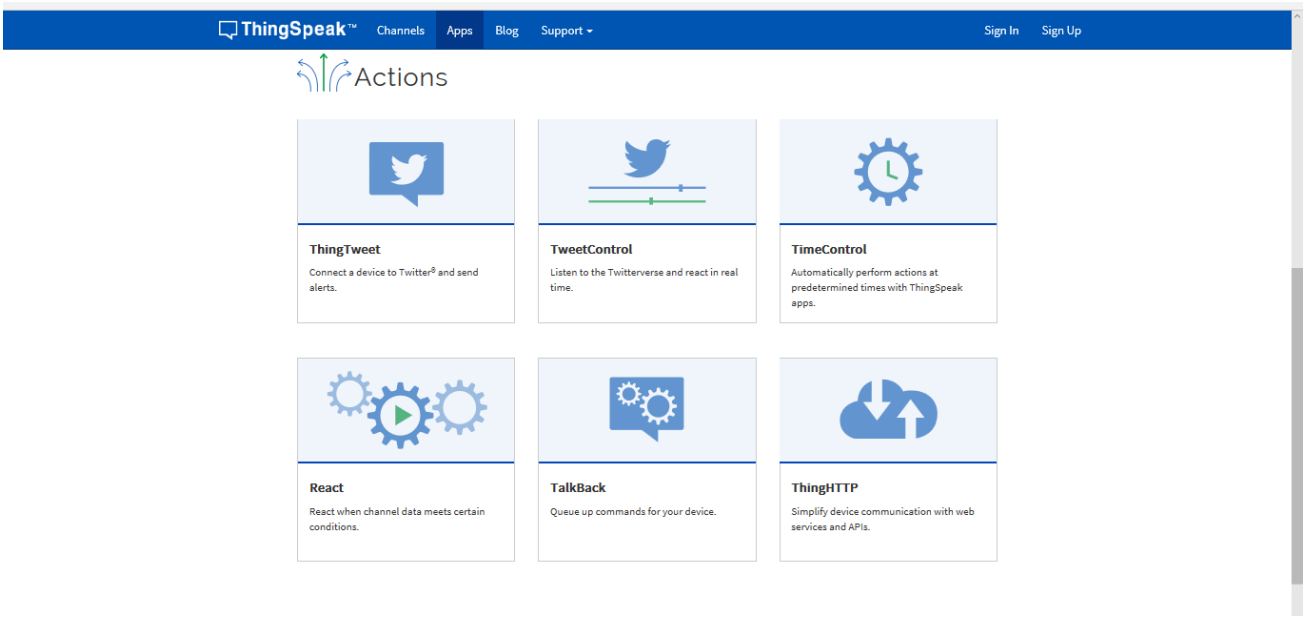
APP: Plugins.

Para extender la funcionalidad del sitio también se nos brinda la oportunidad de desarrollar plugins.

Estos nos ofrecen la posibilidad de crear aplicaciones de forma nativa en nuestra plataforma ThingSpeak.

Soporta HTML, CSS y JavaScript como lenguajes de programación. Al igual que los canales los plugins pueden ser público o privados según sean nuestras necesidades.

Por último, destacar que ofrece la posibilidad de usar Google Gauge Visualización, gracias al cual nos ofrece la posibilidad de visualizar los datos de una forma rápida y amigable, con un nivel de personalización muy amplio.



Sin duda las aplicaciones que encontramos en ThingSpeak son un complemento perfecto para nuestros proyectos, dotándolos en muchos casos de unas funcionalidades muy interesantes, actualmente tienen las siguientes aplicaciones en catálogo.

ThingTweet.

Esta aplicación actúa como proxy permitiendo a nuestro dispositivo enviar actualizaciones de status a Twitter. Esto se consigue gracias a una llamada a la API de ThingTweet.

Twitter da a sus usuarios la opción de actualizar su cuenta a través de una autenticación abierta. Sin embargo, esta opción no está diseñada para dispositivos con capacidades de procesamiento limitadas dichas como Arduino o Netduino. ThingSpeak la API resuelve el problema con ThingTweet APP. Ejecuta un Twitter proxy que envía actualizaciones de estado a Twitter a través de las llamadas ThingSpeak API.

Tweet Control.

Esta aplicación se pone a escuchar los hashtags de Twitter y luego permite controlar cualquier cosa que puedas imaginar. Cualquier cosa que se pueda conectar a través de ThingHTTP también se puede conectar a Twitter.

Permite monitorizar flujos en tiempo real. Hacer esto requiere un servidor dedicado con proceso de larga duración que esté preguntando constantemente a Twitter para las actualizaciones de estado.

ThingControl App hace la tarea mencionada. A continuación, el desarrollador puede concentrarse en determinar el stream y "hasttag" específico y poder realizar una acción de control tal como el envío de una solicitud ThingHTTP a terceras partes.

ThingHTTP.

ThingHTTP permite a un microcontrolado o dispositivo a bajo nivel conectarse con a cualquier servicio web como Prowl y Twilio usando HTTP a través de una red o de Internet. Esta aplicación elimina la necesidad de que los dispositivos implementen el protocolo para tratar con cada servicio web. Además, es posible incorporar un "Parse String" dentro de la petición HTTP a ese servicio, para evitar la codificación de un analizador sintáctico en los dispositivos restringidos.

React.

Básicamente lo que realiza esta app, es ejecutar una acción cuando se cumple una cierta condición para los datos en nuestro canal. Por ejemplo, podríamos encender nuestra cafetera según llegamos a casa del trabajo, creando un **React** de Geolocalización.

La plataforma ThingSpeak procesa streams de datos y permite la posibilidad de establecer acciones de trigger cuando se cumpla una condición con respecto a esos datos en un canal ThingSpeak. Los tipos de condición dependen de los datos específicos a monitorizar; datos de los sensores, textos, etc. Cuando se cumple una condición React puede desencadenar una solicitud ThingHTTP a una tercera entidad o publicar un Tweet con ThingTweet App.

TalkBack.

Permite que cualquier dispositivo pueda actuar sobre los comandos en cola. Por ejemplo, podremos controlar una BOMBA añadiendo comandos tales como ARRANCAR BOMBA Y PARAR BOMBA usando web interface o API. El funcionamiento podría ser el siguiente:

Si se activa una boya la bomba arranca, si la boya se desactiva y permanece en ese estado después de un par de minutos entonces la bomba se para.

TimeControl.

Con TimeControl a un determinada hora, podremos ejecutar un ThingHTTP o un ThingTweet, e incluso añadir un nuevo comando a TalkBack.

5.3.1.3 Integración

Uno de los puntos fuertes en cualquier plataforma IoT, es que permita una amplia integración con diversos dispositivos Hardware y software. En este caso ThingSpeak permite la integración de su plataforma con:

- Arduino.
- Raspberry Pi.
- IoBridge / RealTime.io.
- Electric Imp.
- Móbiles / Aplicaciones web.
- Redes Sociales.
- Análisis de datos con MATLAB.

6 CAPITULO 6: PROTOTIPO.

6.1 RESUMEN DEL PROTOTIPO DISEÑADO.

6.1.1 Objetivo.

Monitorización en tiempo real y archivado de los siguientes parámetros en aguas pluviales en entornos industriales:

1. pH
2. Conductividad
3. Turbidez
4. Temperatura
5. Uso de un Pluviómetro para registro de volumen por área.

El sistema permite añadir aquellos parámetros considerados como relevante en función del entorno industrial de aplicación

6.1.2 Hardware.

Plataforma	Arduino
Ámbito de Aplicación	Prototipos, Educación.
Micro:	Familia Atmel: ATmega328.
Modelo	Arduino Uno R3
Memoria.	32 Kb.
Alimentación	5V-9V.

6.1.3 Plataforma IoT.

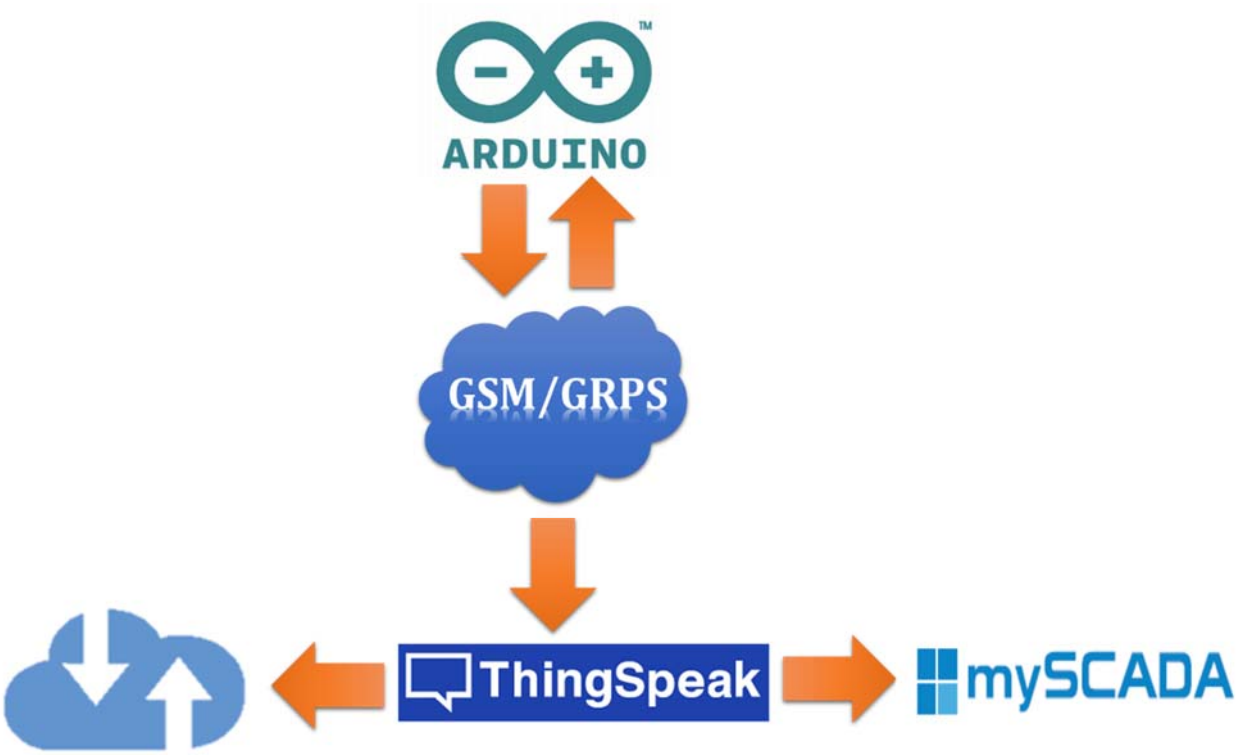
Plataforma	ThingSpeak
Hardware	Arduino, Raspberry Pi, Spark, Electronic Imp.
Ambato	Smart Home, Prototipos.
Ventajas	Interfaz, App, Integración en redes sociales.
Desventajas	Documentación limitada a ciertos HW.

6.2 APLICACIÓN IOT USANDO ARDUINO + THINGSPEAK.

En este punto se va a describir el prototipo desarrollado donde se ha sido implementado todo lo expuesto en los capítulos anteriores, haciendo uso de una placa Arduino UNO rev3 al cuál, le añadimos el Shield GSM GPRS SIM900 para Arduino, para dotarle de comunicación GRPS que nos permita la conexión a Internet con una tarjeta de datos de la compañía SIMIO.

Haremos uso de la plataforma ThingSpeak. La aplicación se basará en recrear un prototipo, basándonos en el área del Smart Sensor, midiendo la temperatura, conductividad, pH, Turbidez y volumen de agua de lluvia en un tanque de tormenta y subiendo los datos medidos a la plataforma ThingSpeak.

Usaremos el app ThingHTTP, para el envío de alarmas por SMS a móviles, de forma, que podamos interactuar con el sistema activando por ejemplo la marcha de una bomba antes de producirse el vertido al cauce público.



6.2.1 Elección Plataformas

Aunque las razones de la elección de plataformas ya han sido explicadas en los capítulos anteriores, a continuación, se detallarán a modo recordatorio porque se ha elegido las plataformas tanto software como hardware para el desarrollo de nuestro prototipo.

El principal motivo para la elección de la plataforma software ThingSpeak, es sin duda su categoría de Open Source, para este tipo de proyecto es sin duda uno de los mejores candidatos tanto a nivel económico como a nivel de documentación.

Otro punto a favor es su interfaz amigable, y de fácil configuración, como además su integración con aplicaciones de terceros véase por ejemplo la red social twitter.

A la hora de elegir que hardware usar, la elección ha sido más sencilla, sin duda Arduino. Plataforma con un gran recorrido y una documentación muy extensa, y una gran comunidad que ayudan a todo aquel que tiene cualquier problema.

Su precio también es un punto a favor, de lo más económico que hay en el mercado.

6.3 FUNCIONALIDAD

Nuestra aplicación realizará las siguientes funciones:

1. Medir diferentes parámetros del agua de lluvia almacenada en un tanque de tormenta mediante los sensores siguientes:
    - Sensor de conductividad.
    - Sensor de turbidez.
    - Sensor de pH.
    - Sensor de Temperatura.
    - Pluviómetro.

Los sensores pH, conductividad y turbidez nos dan información de un posible vertido.

El sensor de temperatura es necesario para realizar las otras mediciones, ya que, por ejemplo, el pH necesita una compensación de temperatura para realizar correctamente la medida.

El pluviómetro nos informa del volumen de lluvia por área.

2. Subir los datos a la plataforma IoT ThingSpeak.

Uso de las aplicaciones ThingHTTP, Tweet Control, React y TalkBack para envío de alarmas a móviles y activación de una bomba sumergible.

Aunque con Tweet Control y ThingHTTP hubiera sido suficiente para activar la bomba, se han usado las demás aplicaciones para mostrar las funcionalidades que trae esta plataforma.

3. Mediante un APP para Android, poder visualizar los datos, ver gráficas, históricos, modificar rangos…, sin acceder directamente a la plataforma ThingSpeak.

6.3.1 Instalación IDE Arduino.

Para poder comenzar a trabajar con nuestra placa Arduino, lo primero que tenemos que hacer es descargarnos el IDE de Arduino, antes de continuar, hay que aclarar que la compañía de Arduino está dividida en dos, arduino.cc y arduino.org, a la hora de tener que descargar el IDE debemos tener esto en cuenta, aunque ambos IDE funcionan perfectamente para ambas placas.

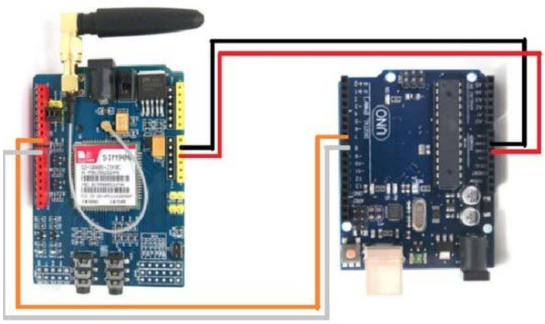
En nuestro caso descargamos el software de la página arduino.org, siendo la versión del IDE actual la 1.7.3.:

http://www.arduino.org/downloads

La instalación es guiada por lo tanto es muy sencilla, una vez instalado conectamos nuestra placa por USB y esperamos a que sea reconocida por nuestro equipo.

6.3.1.1 Conexión GRPS.

Como el modulo nos dispone de pines que permitan la conexión por clip con la placa Arduino, se realizara mediante cableado.



Cables jumper macho-macho

6.3.1.2 Montaje del Circuito

Ya conectado el Shield GSM GPRS SIM900 a nuestra placa Arduino UNO Rev. 3, nos apoyaremos de un protoboard, para la conexión de los sensores.

- Usaremos el pin A0 para el sensor de pH.
- Usaremos el pin A1 para el sensor de Conductividad.
- Usaremos el pin A2 para el sensor de Turbidez.
- Usaremos el pin D2 para el sensor de temperatura.
- Usaremos el pin D4 para el pluviómetro.
- Pin 5V y GND para la alimentación.

6.3.1.3 Descripción y esquema de conexiones

El sistema de sensores final se compone de los cinco sistemas individuales descritos en esta memoria (detección de temperatura, pH, conductividad, turbidez y pluviómetro), que operan de forma independiente, excepto temperatura que forma parte del pH y conductividad, de modo que tanto los errores en el funcionamiento como en el conexionado pueda controlarse y solventarse también de manera independiente. También con el objetivo y para minimizar los errores de conexión, se ha mantenido (en la medida de lo posible) un código de colores en el cableado, desde el diseño inicial de los subsistemas en el software de **Fritzing** hasta el montaje final del hardware, para facilitar el seguimiento y revisión de las señales tal y como se detalla a continuación.

Como ya se ha visto en la sección de cada subsistema, cada sensor se conecta a un pin analógico de Arduino para posteriormente enviarlos a la estación de control.

- pH: Pin analógico A0.
- Conductividad: Pin analógico A1.
- Turbidez Pin analógico A2.
- Temperatura: Pin digital D2.
- Pluviómetro: Pin digital D4

Por otra parte, cada circuito de alarma luminosa tiene su LED correspondiente conectado a un pin digital de la placa Arduino UNO:

Calibración del sensor de pH: LED azul: Pin digital D12. Calibración del sensor de conductividad: LED azul: Pin digital D10.

6.3.2 Código obtención calibración de los sensores.

6.3.2.1 Código pH: Uso del modo analógico.

```
/*************************************************

#include <EEPROM.h>

#define EEPROM_write(address, p) {int i = 0; byte *pp = (byte*)&(p);for(; i < sizeof(p); i++) EEPROM.write(address+i, pp[i]);}

#define EEPROM_read(address, p)  {int i = 0; byte *pp = (byte*)&(p);for(; i < sizeof(p); i++) pp[i]=EEPROM.read(address+i);}


#define ReceivedBufferLength 20

char receivedBuffer[ReceivedBufferLength+1];   // store the serial command

byte receivedBufferIndex = 0;


#define SCOUNT  30           // sum of sample point

int analogBuffer[SCOUNT];    //store the sample voltage

int analogBufferIndex = 0;


#define SlopeValueAddress 0     // (slope of the ph probe)store at the beginning of the EEPROM. The slope is a float number,occupies 4 bytes.

#define InterceptValueAddress (SlopeValueAddress+4)

float slopeValue, interceptValue, averageVoltage;

boolean enterCalibrationFlag = 0;


#define SensorPin A0

#define VREF 5000  //for arduino uno, the ADC reference is the power(AVCC), that is 5000mV


void setup()

{

 Serial.begin(115200);

 readCharacteristicValues(); //read the slope and intercept of the ph probe
```

```
}


void loop()
{
 if(serialDataAvailable() > 0)
 {
    byte modeIndex = uartParse();
    phCalibration(modeIndex);    // If the correct calibration command is received, the calibration function should be called.
    EEPROM_read(SlopeValueAddress, slopeValue);    // After calibration, the new slope and intercept should be r ead ,to update current value.
    EEPROM_read(InterceptValueAddress, interceptValue);
 }


  static unsigned long sampleTimepoint = millis();
  if(millis()-sampleTimepoint>40U)
  {
   sampleTimepoint = millis();
   analogBuffer[analogBufferIndex] = analogRead(SensorPin)/1024.0*VREF;    //read the voltage and store into t he buffer,every 40ms
   analogBufferIndex++;
   if(analogBufferIndex == SCOUNT)
      analogBufferIndex = 0;
   averageVoltage = getMedianNum(analogBuffer,SCOUNT);   // read the stable value by the median filtering algor ithm
  }


  static unsigned long printTimepoint = millis();
  if(millis()-printTimepoint>1000U)
  {
   printTimepoint = millis();
   if(enterCalibrationFlag)          // in calibration mode, print the voltage to user, to watch the stability of voltage
   {
```

```
      Serial.print("Voltage:");
      Serial.print(averageVoltage);
      Serial.println("mV");
    }else{
    Serial.print("pH:");          // in normal mode, print the ph value to user
    Serial.println(averageVoltage/1000.0*slopeValue+interceptValue);
    }
  }
}

boolean serialDataAvailable(void)
{
 char receivedChar;
 static unsigned long receivedTimeOut = millis();
 while (Serial.available()>0)
 {
  if (millis() - receivedTimeOut > 1000U)
  {
   receivedBufferIndex = 0;
   memset(receivedBuffer,0,(ReceivedBufferLength+1));
  }
  receivedTimeOut = millis();
  receivedChar = Serial.read();
  if (receivedChar == '\n' || receivedBufferIndex==ReceivedBufferLength){
                        receivedBufferIndex = 0;
                        strupr(receivedBuffer);
                        return true;
  }
  else{
   receivedBuffer[receivedBufferIndex] = receivedChar;
   receivedBufferIndex++;
```

```
  }
 }
 return false;
}

byte uartParse()
{
 byte modeIndex = 0;
 if(strstr(receivedBuffer, "CALIBRATION") != NULL)
    modeIndex = 1;
 else if(strstr(receivedBuffer, "EXIT") != NULL)
    modeIndex = 4;
 else if(strstr(receivedBuffer, "ACID:") != NULL)
    modeIndex = 2;
 else if(strstr(receivedBuffer, "ALKALI:") != NULL)
    modeIndex = 3;
 return modeIndex;
}

void phCalibration(byte mode)
{
  char *receivedBufferPtr;
  static byte acidCalibrationFinish = 0, alkaliCalibrationFinish = 0;
  static float acidValue,alkaliValue;
  static float acidVoltage,alkaliVoltage;
  float acidValueTemp,alkaliValueTemp,newSlopeValue,newInterceptValue;
  switch(mode)
  {
   case 0:
   if(enterCalibrationFlag)
     Serial.println(F("Command Error"));
```

```
    break;

    case 1:
    receivedBufferPtr=strstr(receivedBuffer, "CALIBRATION");
    enterCalibrationFlag = 1;
    acidCalibrationFinish = 0;
    alkaliCalibrationFinish = 0;
    Serial.println(F("Enter Calibration Mode"));
    break;

    case 2:
    if(enterCalibrationFlag)
    {
       receivedBufferPtr=strstr(receivedBuffer, "ACID:");
       receivedBufferPtr+=strlen("ACID:");
       acidValueTemp = strtod(receivedBufferPtr,NULL);
       if((acidValueTemp>3)&&(acidValueTemp<5))       //typical ph value of acid standand buffer solution should
be 4.00
       {
         acidValue = acidValueTemp;
         acidVoltage = averageVoltage/1000.0;       // mV -> V
         acidCalibrationFinish = 1;
         Serial.println(F("Acid Calibration Successful"));
       }else {
         acidCalibrationFinish = 0;
         Serial.println(F("Acid Value Error"));
       }
    }
    break;

     case 3:
```

```
    if(enterCalibrationFlag)
    {
      receivedBufferPtr=strstr(receivedBuffer, "ALKALI:");
      receivedBufferPtr+=strlen("ALKALI:");
      alkaliValueTemp = strtod(receivedBufferPtr,NULL);
      if((alkaliValueTemp>8)&&(alkaliValueTemp<11))      //typical ph value of alkali standand buffer solution s
hould be 9.18 or 10.01
      {
         alkaliValue = alkaliValueTemp;
         alkaliVoltage = averageVoltage/1000.0;
         alkaliCalibrationFinish = 1;
         Serial.println(F("Alkali Calibration Successful"));
       }else{
        alkaliCalibrationFinish = 0;
        Serial.println(F("Alkali Value Error"));
       }
    }
    break;

     case 4:
     if(enterCalibrationFlag)
     {
       if(acidCalibrationFinish && alkaliCalibrationFinish)
       {
        newSlopeValue = (acidValue-alkaliValue)/(acidVoltage - alkaliVoltage);
        EEPROM_write(SlopeValueAddress, newSlopeValue);
        newInterceptValue = acidValue - (slopeValue*acidVoltage);
        EEPROM_write(InterceptValueAddress, newInterceptValue);
        Serial.print(F("Calibration Successful"));
       }
       else Serial.print(F("Calibration Failed"));
```

```
        Serial.println(F(",Exit Calibration Mode"));
        acidCalibrationFinish = 0;
        alkaliCalibrationFinish = 0;
        enterCalibrationFlag = 0;
      }
      break;
   }
}

int getMedianNum(int bArray[], int iFilterLen)
{
    int bTab[iFilterLen];
    for (byte i = 0; i<iFilterLen; i++)
    {
                bTab[i] = bArray[i];
    }
    int i, j, bTemp;
    for (j = 0; j < iFilterLen - 1; j++)
    {
                for (i = 0; i < iFilterLen - j - 1; i++)
       {
                 if (bTab[i] > bTab[i + 1])
         {
                            bTemp = bTab[i];
                   bTab[i] = bTab[i + 1];
                            bTab[i + 1] = bTemp;
                  }
                }
    }
    if ((iFilterLen & 1) > 0)
              bTemp = bTab[(iFilterLen - 1) / 2];
```

```
    else
            bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
    return bTemp;
}

void readCharacteristicValues()
{
  EEPROM_read(SlopeValueAddress, slopeValue);
  EEPROM_read(InterceptValueAddress, interceptValue);
  if(EEPROM.read(SlopeValueAddress)==0xFF && EEPROM.read(SlopeValueAddress+1)==0xFF && EEPROM.rea
d(SlopeValueAddress+2)==0xFF && EEPROM.read(SlopeValueAddress+3)==0xFF)
  {
   slopeValue = 3.5;   // If the EEPROM is new, the recommendatory slope is 3.5.
   EEPROM_write(SlopeValueAddress, slopeValue);
  }
  if(EEPROM.read(InterceptValueAddress)==0xFF && EEPROM.read(InterceptValueAddress+1)==0xFF && EEPR
OM.read(InterceptValueAddress+2)==0xFF && EEPROM.read(InterceptValueAddress+3)==0xFF)
  {
   interceptValue = 0;  // If the EEPROM is new, the recommendatory intercept is 0.
   EEPROM_write(InterceptValueAddress, interceptValue);
```

6.3.2.2 Código Conductividad: Uso del modo analógico.

```
// # Nombre Producto: Analog EC Meter
// # Modelo SKU : DFR0300
// # Version    : 1.0

// # Connection:
// #       EC meter output    -> Analog pin 1
// #       DS18B20 digital pin -> Digital pin 2
// #

#include <OneWire.h>

#define StartConvert 0
#define ReadTemperature 1

const byte numReadings = 20;     //the number of sample times
byte ECsensorPin = A1;  //EC Meter analog output,pin on analog 1
byte DS18B20_Pin = 2; //DS18B20 signal, pin on digital 2
unsigned int AnalogSampleInterval=25,printInterval=700,tempSampleInterval=850;  //analog sample interval;ser
ial print interval;temperature sample interval
unsigned int readings[numReadings];      // the readings from the analog input
byte index = 0;                  // the index of the current reading
unsigned long AnalogValueTotal = 0;                  // the running total
unsigned int AnalogAverage = 0,averageVoltage=0;                // the average
unsigned long AnalogSampleTime,printTime,tempSampleTime;
float temperature,ECcurrent;

//Temperature chip i/o
OneWire ds(DS18B20_Pin);  // on digital pin 2

void setup() {
```

```
 // initialize serial communication with computer:
  Serial.begin(115200);
  // initialize all the readings to 0:
  for (byte thisReading = 0; thisReading < numReadings; thisReading++)
    readings[thisReading] = 0;
  TempProcess(StartConvert);   //let the DS18B20 start the convert
  AnalogSampleTime=millis();
  printTime=millis();
  tempSampleTime=millis();
}

void loop() {
  /*
   Every once in a while,sample the analog value and calculate the average.
  */
  if(millis()-AnalogSampleTime>=AnalogSampleInterval)
  {
    AnalogSampleTime=millis();
     // subtract the last reading:
    AnalogValueTotal = AnalogValueTotal - readings[index];
    // read from the sensor:
    readings[index] = analogRead(ECsensorPin);
    // add the reading to the total:
    AnalogValueTotal = AnalogValueTotal + readings[index];
    // advance to the next position in the array:
    index = index + 1;
    // if we're at the end of the array...
    if (index >= numReadings)
    // ...wrap around to the beginning:
    index = 0;
    // calculate the average:
```

```
  AnalogAverage = AnalogValueTotal / numReadings;
 }
 /*
  Every once in a while,MCU read the temperature from the DS18B20 and then let the DS18B20 start the convert.
  Attention:The interval between start the convert and read the temperature should be greater than 750 milliseco
nd,or the temperature is not accurate!
 */
  if(millis()-tempSampleTime>=tempSampleInterval)
 {
   tempSampleTime=millis();
   temperature = TempProcess(ReadTemperature);  // read the current temperature from the  DS18B20
   TempProcess(StartConvert);                   //after the reading,start the convert for next reading
 }
  /*
  Every once in a while,print the information on the serial monitor.
 */
 if(millis()-printTime>=printInterval)
 {
   printTime=millis();
   averageVoltage=AnalogAverage*(float)5000/1024;
   Serial.print("Analog value:");
   Serial.print(AnalogAverage);   //analog average,from 0 to 1023
   Serial.print("    Voltage:");
   Serial.print(averageVoltage);  //millivolt average,from 0mv to 4995mV
   Serial.print("mV    ");
   Serial.print("temp:");
   Serial.print(temperature);    //current temperature
   Serial.print("^C     EC:");


   float TempCoefficient=1.0+0.0185*(temperature-25.0);    //temperature compensation formula: fFinalResult(2
5^C) = fFinalResult(current)/(1.0+0.0185*(fTP-25.0));
   float CoefficientVolatge=(float)averageVoltage/TempCoefficient;
```

```
  if(CoefficientVolatge<150)Serial.println("No solution!");   //25^C 1413us/cm<-->about 216mv  if the voltage(co
mpensate)<150,that is <1ms/cm,out of the range
  else if(CoefficientVolatge>3300)Serial.println("Out of the range!");  //>20ms/cm,out of the range
  else
  {
   if(CoefficientVolatge<=448)ECcurrent=6.84*CoefficientVolatge-64.32;   //1ms/cm<EC<=3ms/cm
   else if(CoefficientVolatge<=1457)ECcurrent=6.98*CoefficientVolatge-127;  //3ms/cm<EC<=10ms/cm
   else ECcurrent=5.3*CoefficientVolatge+2278;                  //10ms/cm<EC<20ms/cm
   ECcurrent/=1000;    //convert us/cm to ms/cm
   Serial.print(ECcurrent,2);  //two decimal
   Serial.println("ms/cm");
  }
 }


}
/*
ch=0,let the DS18B20 start the convert;ch=1,MCU read the current temperature from the DS18B20.
*/
float TempProcess(bool ch)
{
 //returns the temperature from one DS18B20 in DEG Celsius
 static byte data[12];
 static byte addr[8];
 static float TemperatureSum;
 if(!ch){
      if ( !ds.search(addr)) {
         Serial.println("no more sensors on chain, reset search!");
         ds.reset_search();
         return 0;
      }
      if ( OneWire::crc8( addr, 7) != addr[7]) {
```

```
        Serial.println("CRC is not valid!");
        return 0;
      }
      if ( addr[0] != 0x10 && addr[0] != 0x28) {
        Serial.print("Device is not recognized!");
        return 0;
      }
      ds.reset();
      ds.select(addr);
      ds.write(0x44,1); // start conversion, with parasite power on at the end
 }
 else{
      byte present = ds.reset();
      ds.select(addr);
      ds.write(0xBE); // Read Scratchpad
      for (int i = 0; i < 9; i++) { // we need 9 bytes
       data[i] = ds.read();
      }
      ds.reset_search();
      byte MSB = data[1];
      byte LSB = data[0];
      float tempRead = ((MSB << 8) | LSB); //using two's compliment
      TemperatureSum = tempRead / 16;
   }
      return TemperatureSum;
}
```

6.3.2.3 Código Turbidez: Uso del modo analógico.

```
void setup() {
  Serial.begin(9600); //Baud rate: 9600
}
void loop() {
  int sensorValue = analogRead(A0);// read the input on analog pin 0:
  float voltage = sensorValue * (5.0 / 1024.0); // Convert the analog reading (which goes from 0 - 1023) to a voltag
e (0 - 5V):
  Serial.println(voltage); // print out the value you read:
  delay(500);
}
```

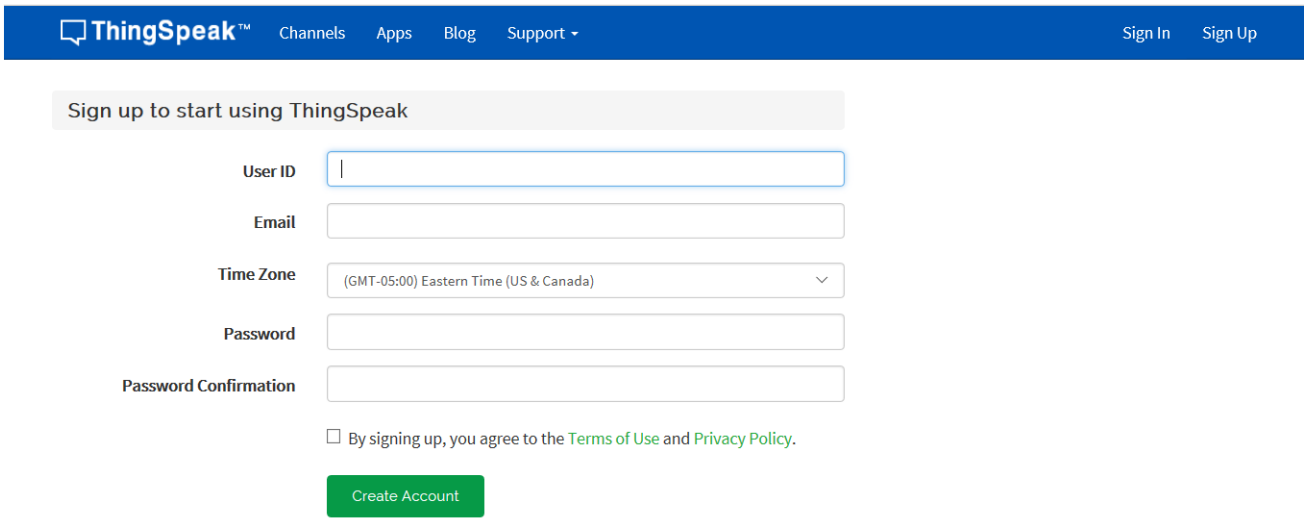
6.3.3 Plataforma ThingSpeak

Una vez montada la parte hardware pasaremos a configurar nuestra plataforma software.

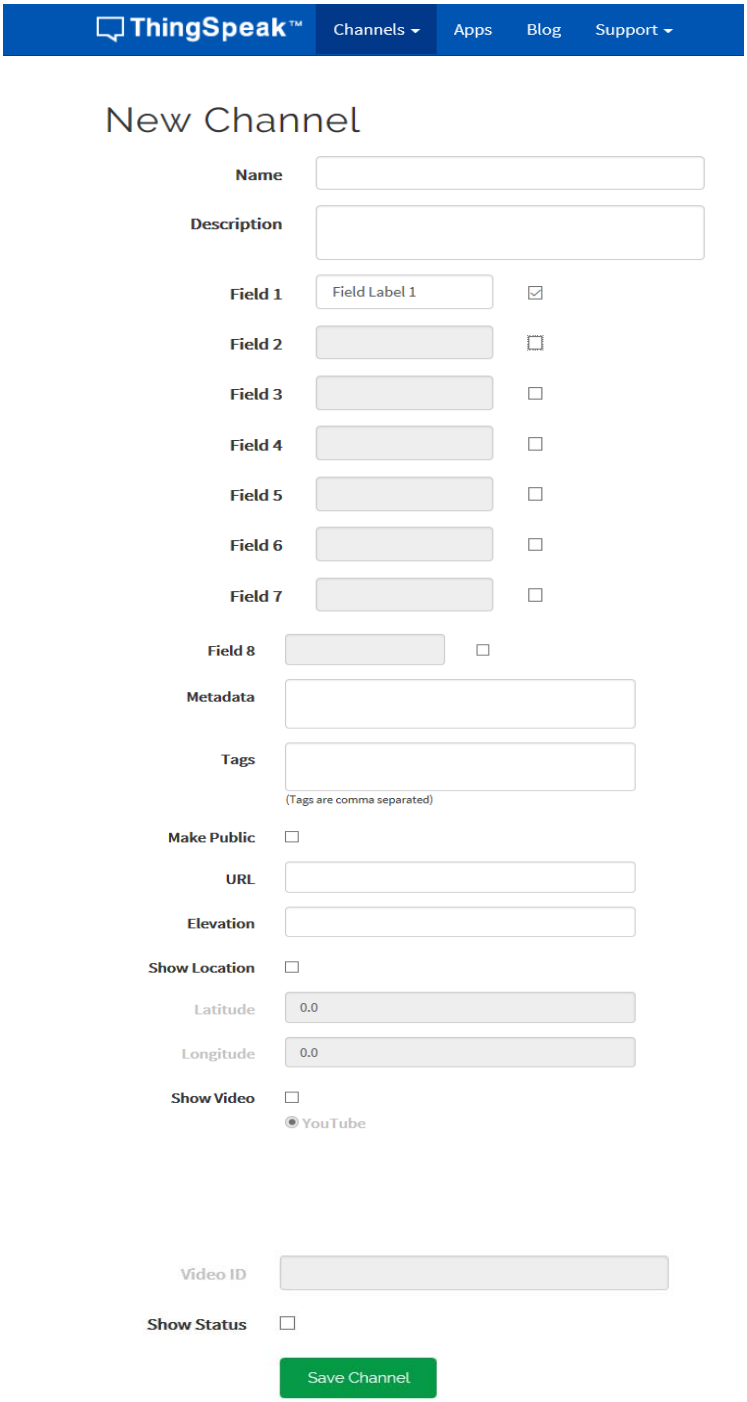
Lo primero que tenemos que hacer es darnos de alta en la plataforma, para ello entramos en la página:

https://thingspeak.com/login

En el vínculo Sing up nos damos de alta.



Una vez dado de alta, nos vamos a la pestaña de Chanels y creamos nuestros canales, uno por cada sensor que deseemos registrar, a continuación rellenamos los siguientes campos:



Autor: Inmaculada González

Con esto ya tenemos creado los canales donde se volcarán las medidas registradas por nuestros sensores, un dato importante que tenemos que anotar para poder subir los datos es la WRITE API KEY, la cual a la hora de programar nuestro Arduino deberemos incluirla para poder subir los datos a la plataforma

.**API ThingHTTP.**

Configuraremos esta aplicación para enviar alarmas a móviles cuando alguno de los parámetros supere la consigna establecida.

6.4 SOFTWARE MYSCADA.

Para la visualización de los parámetros monitorizados y sus gráficas, se ha desarrollado una aplicación mediante el software gratuito para uso no comercial, denominado My Scada.

6.4.1 Descripción.

My SCADA es un sistema de control con todas las funciones y aplicaciones para la Adquisición y manejo de Datos (Sistemas SCADA) con una Interfaz avanzada hombre-máquina o también llamada HMI. El software proporciona al usuario la capacidad de controlar, monitorear y mostrar el estado de los procesos de tipo:

- *Tecnológico*
- *Infraestructura*
- *Sistemas Industriales*
- *Instalaciones*

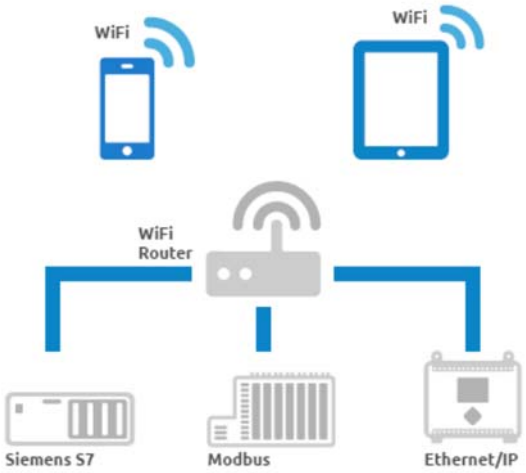
La aplicación está diseñada para comunicarse con los dispositivos de Rockwell Automation con una red Ether-Net / IP, permite usar el protocolo Modbus TCP Unidad de terminal remota (RTU) con los dispositivos de control desarrollados por las empresas como Schneider, Delta, Wago, Siemens, Advantech, Unitronics, Beckhoff, etc, con el apoyo del protocolo Modbus, puede integrar directamente un gran número de sensores y dispositivos I / O. Además se puede comunicar con protocolo MTTQ y con JavaScript con el servidor de ThingSpeak

El editor de My SCADA es un entorno de desarrollo integrado que se utiliza para configurar, desarrollar y gestionar ventanas HMI.En la aplicación se encontrará todo lo necesario para implementar una

herramienta completa SCADA. El editor de My SCADA se utiliza para crear y gestionar proyectos, configurar conexiones a otros dispositivos, entre las etiquetas, las alarmas y tendencias**.**

Una de las ventajas de My scada es que es multiplataforma, es decir, se desarrolla con una sola aplicación y se puede correr en Windows, Android, IOS, Linux

My SCADA también es un sistema multiprotocolo y cubre la mayoría de los PLC conocidos o RTUs y sus protocolos para mantener la versatilidad.
Soporta la mayoría de protocolos de la industria:



Ejemplo de pantallas desarrollas.

Autor: Inmaculada González

Autor: Inmaculada González

6.4.2 Ámbitos de aplicación.

Este prototipo además de monitorizar parámetros sobre la calidad del agua podría tener otras aplicaciones sin variar mucho el código actual como la monitorización de emisiones de gases y partículas a la atmósfera.

Para ello se utilizaría el mismo prototipo de Arduino y la plataforma ThingSpeak y únicamente se cambiarían los sensores, ya que, se utilizarían sensores para control de emisiones, tales como:

Sensor de CO2
Sensor de NOx
Sensor de CH4
Sensor de VOC
Sensor de partículas en suspensión

6.5 PRESUPUESTO DEL PROYECTO.

Estamos ante un proyecto el desarrollo de un prototipo académico, por lo que el coste distaría bastante de lo que sería un proyecto real a nivel corporativo, ya que, los sensores a utilizar aunque de bajo coste se pondrían de una marca resistente a entornos industriales, en el apartado del Capítulo II, se describen un tipo que podrían ser usados por su compatibilidad con el Hardware Arduino.

El presupuesto descrito a continuación detalla el coste que supondría el prototipo con sensores de la marca considerada

1. Ingeniería:
2. Proyecto.
3. Desarrollos del Código.
4. Testeo y depuración del código
5. Equipos
6. Arduino
7. Sensores.
8. Material Auxiliar
9. Montaje
10. Puesta en servicio.

6.5.1 Comparativa con sistemas tradicionales.

Por último se muestra una comparativa a nivel coste entre el prototipo y un sistema tradicional.

7 MANUALES DE LOS SENSORES.

8 ANEXO IV: GLOSARIO.

Android: Sistema operativo basado en Linux utilizado para móviles.

Arduino: Plataforma libre que consiste en una placa, un micro-controlador y un entorno de desarrollo que facilita el uso de electrónica.

Framework: Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

HTML: Lenguaje de marcado predominante para la creación de sitios web. Es un estándar...

HTML5: Versión 5 del estándar HTML.

HTTP: Protocolo de transferencia de hipertexto usando en las transacciones de la World Wide Web (WWW).

Internet of Things: Red inteligente que contiene objetos heterogéneos y ubicuos interactuando entre ellos.

Java: Lenguaje de programación de alto nivel orientado a objetos creado por Sun Microsystems.

JavaScript: Lenguaje de programación interpretado y basado en prototipos. Es un dialecto del estándar ECMAScript.

JSON: Formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. MIDGAR: Plataforma para la generación dinámica de aplicaciones distribuidas basadas en la integración de redes de sensores y dispositivos electrónicos IOT

Lenguaje de Dominio Específico: Especificación de un lenguaje dedicado a resolver un problema en particular, representar un problema específico y proveer una técnica para solucionar una situación particular.

Micro-controlador: Circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria.

GPRS significa General Packet Radio Service o en español Servicio General de Paquetes por Radio.

RFID: Sistema de almacenamiento y recuperación de datos remotos que usan dispositivos denominados etiquetas, tarjetas, transpondedores o tags RFID. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio. Las tecnologías RFID se agrupan dentro de las denominadas Auto ID (Automatic IDentification, o identificación automática). Son unos dispositivos pequeños, similares a una pegatina, que pueden ser adheridas o incorporadas a un producto, un animal o una persona

Ruby on Rails: Framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible. Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby.

Sensor: Dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, movimiento, pH, etc. Una magnitud eléctrica puede ser una resistencia eléctrica (como en una RTD), una capacidad eléctrica (como en un sensor de humedad), una Tensión eléctrica (como en un termopar), una corriente eléctrica (como en un fototransistor), etc.

SQL: Lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ella.

Smart Home: Engloba la automatización de la actividad en el hogar y las tareas domésticas. La domótica puede incluir un control centralizado de la iluminación, HVAC (calefacción, ventilación y aire acondicionado), los electrodomésticos, y otros sistemas, para proporcionar mayor comodidad, el confort, la eficiencia energética y la seguridad.

Smart Object: Elemento físico, con diversas propiedades, identificable a lo largo de su vida útil, que interactúa con el entorno y otros objetos y que puede actuar de manera inteligente según unas determinadas situaciones, mediante una conducta autónoma.

Smartphone: Teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar actividades semejantes a una mini computadora y conectividad que un teléfono móvil convencional. El término «inteligente» hace referencia a la capacidad de usarse como un ordenador de bolsillo, llegando incluso a remplazar a un ordenador personal en algunos casos.

World Wide Web: Sistema de distribución de información basado en hipertexto o hipermedios enlazados y accesibles a través de Internet. Con un navegador web, un usuario visualiza sitios web MIDGAR: Plataforma para la generación dinámica de aplicaciones distribuidas basadas en la integración de redes de sensores y dispositivos electrónicos compuestos de páginas web que pueden contener texto, imágenes, vídeos u otros contenidos multimedia, y navega a través de esas páginas usando hiperenlaces.

XML: Lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes.

IDE: Entorno de desarrollo integrado. Interfaz de programación.

NetBeans: Es un IDE, desarrollado principalmente para el lenguaje de programación

Java.

API: Interfaz de programación de aplicaciones. Es un conjunto de funciones y

Procedimientos. Ofrece cierta biblioteca de la cual se puede tomar dichas funciones o

Procedimientos.

GSM: Sistema Global para las telecomunicaciones móviles.

Gateway: Puerta de enlace. Permite interconectar redes con protocolos y arquitecturas

diferentes.

LAN: Red de Área Local.

WAM: Red de área metropolitana.

GPL: Licencia publica general. Licencia de software libre.

C: Lenguaje de programación orientado a la implementación de sistemas operativos.

AMRA: Analysis and Monitoring of Environmental Risk.

SIG: Sistema de Información Geográfica.

SSL: Secure Sockets Layer.

GNU LGPL: Lesser General Public License (Licencia Pública General Reducida)

WSAN Wireless sensor and actor networks MIDGAR: Plataforma para la generación dinámica de aplicaciones distribuidas basadas en la integración de redes de sensores y dispositivos electrónicos IOT

8.1 ACRONIMOS.

HTML HyperText Markup Language

HTTP Hyptertext Transfer Protocol

IoT Internet of Things

IoTU Usuarios interesados en Internet of Things

JSON JavaScript Object Notation

MDE Model-driven engineering

MOF Meta-Object Facility

NFC Near field communication

NoSQL No Structured Query Language

OMG Object Manager Group

PIM Platform-Independent Model

PSM Platform-Specific Model

REST Representational State Transfer

RFID Radio Frequency IDentification

SDev Software Developers

SIoT Social Internet of Things

SOA Service-oriented architecture

SQL Structured Query Language

UML Unified Modeling Language

WWW World Wide Web.

XMI XML Metadata Interchange

XML eXtensible Markup Language

9 BIBLIOGRAFIA COSULTADA:

1. Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, *54*(15), 2787–2805. doi:10.1016/j.comnet.2010.05.010
2. Hribernik, K. A., Ghrairi, Z., Hans, C., & Thoben, K. (2011). Co-creating the Internet of Things - First Experiences in the Participatory Design of Intelligent Products with Arduino (pp. 1–9).
3. IoBridge. (2013). Thingspeak. Retrieved June 23, 2013, from http://www.thingspeak.com
4. McFarlane, D., Sarma, S., Chirn, J. L., Wong, C . . ., & Ashton, K. (2003). Auto ID systems and intelligent manufacturing control. *Engineering Applications of Artificial Intelligence*, *16*(4), 365–376. Doi: 10.1016/S0952-1976(03)00077-.
5. Juan Andrés Sánchez Wevar, Análisis y Estudio de Redes GPRS, 2005, Trabajo de Titulación para optar al Título de Ingeniero Electrónico.
6. Massimo Banzi, Tom Igoe, Gianluca Martino, and David Mellis. (2013, Noviembre) Arduino. [Online]. (Recuperado: 20 de marzo del 2014).
7. http://arduino.cc/es/#.Uw1iIYXGVSE
8. http://postscapes.com/internet-of-things-software-guide#protocols
9. http://www.datamation.com/open-source/35-open-source-tools-for-the-internet-of-things-1.html
10. https://thingspeak.com/
11. http://www.thinkingthings.telefonica.com/
12. http://www.zatar.com/
13. https://es.wikipedia.org/
14. http://www.libelium.com/es
15. http://www.arduino.cc/
16. http://www.intel.es/content/
17. http://www.zigbee.org/
18. http://www.instructables.com/id/Arduino-Tutorial-0-Introducci%C3%B3n/step3/Qu%C3%A9-partes-componen-el-Arduino

10 ANEXO I. MANUAL DE USUARIO DE ARDUINO.

11 ANEXO II. MANUAL DE USUARIO DE THINKSPEAK

12 ANEXO III. MANUAL DE USUARIO DE LA APLICACIÓN MÓVIL.

13 ANEXO IV. CÓDIGO ARDUINO.