INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

INGENIERIA ELECTRÓNICA

RESIDENCIA PROFESIONAL

DISEÑO DE UN PROTOTIPO DE MONITOREO DE UN INVERNADERO

ASESORES:

ING. ODILIO OROZCO MAGDALENO

DR. HECTOR RICARDO HERNANDEZ DE LEON

ALUMNOS:

ROGER PEÑA HERNANDEZ

10270094

JOSE FRANCISCO MEZA ZAVALETA

10270084

TUXTLA GUTIERREZ, CHIAPAS, MEXICO DEL 2015.

ÍNDICE

CAPÍTULO 1

GENERALIDADES

1.1 INTRODUCCIÓN

El presente documento tiene como finalidad describir el diseño de un sistema de adquisición de datos para el monitoreo y diagnóstico de un invernadero mediante una página web, recopilando la mayor cantidad de datos posibles para así poder procesarlos, particularmente son la temperatura, humedad, CO2, luminosidad.

Estos datos hacen posible saber el comportamiento de los parámetros ambientales en el invernadero, sin embargo la naturaleza cambiante del planeta puede hacer que varíen de un momento a otro durante la medición.

La medición de estas variables se hace por medio de una estación meteorológica, automatizada y diseñada especialmente para el invernadero; es muy importante que los sensores que hacen la captura de los datos sean sensibles y duraderos.

1.2 INFORMACIÓN DE LA INSTITUCIÓN DONDE SE DESARROLLÓ EL PROYECTO

1.2.1 HISTORIA DEL ITTG

El Instituto Tecnológico de Tuxtla Gutiérrez es una institución pública dependiente de la Secretaría de Educación Pública. Imparte 8 licenciaturas y 2 programas de posgrado en las áreas de Ingeniería, Tecnología y Ciencias Administrativas.

El Instituto Tecnológico de Tuxtla Gutiérrez fue fundado el 24 de octubre de 1972, se encuentra ubicado en el centro del estado de Chiapas, que por su gran variedad de climas y suelos es propio para el cultivo de muy diversas especies vegetales nativas y adaptadas, con lo que se puede desarrollar la agroindustria, así como muchos otros procesos industriales a partir del gran potencial que ofrece este estado, en el cual el Instituto debe constituirse en el sujeto de cambio, al presentar alternativas de desarrollo sustentable del estado en forma multidisciplinaria.

Los principales laboratorios con los que cuenta el Instituto Tecnológico de Tuxtla Gutiérrez son:

- ✓ Microbiología
- ✓ Biotecnología
- ✓ Química
- ✓ Química pesada
- ✓ Mecánica
- ✓ Sistemas computacionales
- ✓ Ingeniería industrial
- ✓ Plantas piloto
- ✓ Polo Tecnológico Nacional

1.2.2 MISIÓN

Formar de manera integral profesionistas de excelencia en el campo de la ciencia y la tecnología con actitud emprendedora, respeto al medio ambiente y apego a los valores éticos.

1.2.3 VISIÓN

Ser una Institución de excelencia en la educación superior tecnológica del Sureste, comprometida con el desarrollo socioeconómico sustentable de la región.

1.2.4 LOCALIZACIÓN

Figura 1.1: Carretera Panamericana Km 1080. Tuxtla Gutiérrez Chiapas, C.P. 29050.

1.2.5 ÁREA ESPECÍFICA RELACIONADA DIRECTAMENTE CON EL PROYECTO

El laboratorio de Ingeniería Electrónica cuenta con 8 áreas destinadas a la investigación, desarrollo e implementación de proyectos escolares, 5 de las cuales son educativas, en las que los docentes imparten clases de: electrónica digital, PLC, electrónica de potencia, instrumentación, mediciones eléctricas, programación en c y electrónicas analógicas. Dos más son para investigación, una pertenece al grupo de IEEE y la otra es de desarrollo de circuitos electrónicos. La última área está destinada para que los alumnos desarrollen sus actividades en extra clase. Parte del área que se ocupó en la realización de éste proyecto fueron las de IEEE, las de Electrónicas Analógicas, desarrollo de circuitos electrónicos y el cubículo IEEE.

La carrera de Ingeniería Electrónica tiene como misión “formar profesionales de excelencia con competencias en el ámbito de la Ingeniería Electrónica, motivados para la promoción del desarrollo profesional y el conocimiento científico y tecnológico, con actitud emprendedora, respeto al medio ambiente y apego a los valores cívicos y éticos”.

1.3 ANTECEDENTES

Invernaderos.

Los invernaderos cuentan con una superficie de 600 metros cuadrados, cuenta con servicio de agua, cuarto de herramientas y un espacio de 200 metros cuadrados protegidos con malla sombra.

FIGURA 1.2 EJEMPLO DE INVERNADEROS

1.4 PLANTEMIENTO DEL PROBLEMA

El Instituto Tecnológico de Tuxtla Gutiérrez cuenta con 4 invernaderos dentro de sus instalaciones, los cuales sirven para el cultivo y producción de plantíos de diferentes clases, los invernaderos son monitoreados de forma presencial, es decir, los encargados tienen que estar monitoreando de manera aleatoria físicamente el invernadero, deben estar presentes dentro del invernadero para poder monitorear el ambiente que tienen a ciertas horas los invernaderos para que así puedan hacer las correcciones correspondientes y de esa manera el desarrollo del cultivo no tenga consecuencias negativas.

1.5 OBJETIVOS

1.5.1 OBJETIVO GENERAL:

Diseñar un prototipo de monitoreo automático del ambiente de un invernadero y generar una base de datos.

1.5.2 OBJETIVOS ESPECÍFICOS:

1. Diseñar y construir una tarjeta electrónica para evaluar las variables caracterizadas.
2. Implementar y desarrollar algoritmo de comunicación Ethernet.
3. Monitorear a través de protocolo vía Ethernet.
4. Generar una base de datos de la información obtenida.

1.6 JUSTIFICACIÓN

La razón que nos motivó para diseñar este proyecto es que todas las personas en nuestro planeta están interesadas, de alguna manera u otra en el medio ambiente y los factores que se encuentran en él, ésta es una razón importante para la construcción de una estación meteorológica que sea monitoreada vía remota desde una página web, su construcción es de gran utilidad dado que por medio de éste conoceremos los parámetros de temperatura, humedad, luminosidad, CO2, de esta manera los usuarios siempre podrán monitorear el invernadero de manera remota sin tener que estar presentes necesariamente incluso en el mismo instituto.

Es necesario conocer todos estos parámetros ya que tener el conocimiento de estos parámetros brindará la ayuda a los bioquímicos en detectar estos factores ambientales y así puedan tomar precauciones necesarias para evitar daños en el invernadero del instituto.

Con el proyecto **Diseño de un prototipo de monitoreo para un invernadero** que estará ubicado en el invernadero del instituto tecnológico de Tuxtla Gutiérrez, se harán mediciones de temperatura, humedad, luminosidad, y CO2 durante todo el día (24 horas) con los datos recabados se realizara una base de datos para la representación de su comportamiento durante la adquisición de datos.

1.7 ALCANCES Y LIMITACIONES

1.7.1 ALCANCES:

En lo generan este proyecto permite el monitoreo de temperatura, humedad, luminosidad y CO2 por medio de IP, esto permitirá que se pueda observar el comportamiento del medio ambiente mediante una página web. Por lo que el o los usuarios podrán monitorear el invernadero desde el lugar donde estén, tan solo necesitarán un punto de acceso a internet para poder hacerlo. Además los datos recabados por la estación meteorológica instalada dentro de uno de los invernaderos serán guardados en una PC haciendo así una base de datos para su análisis por horas, días o como lo decida el usuario.

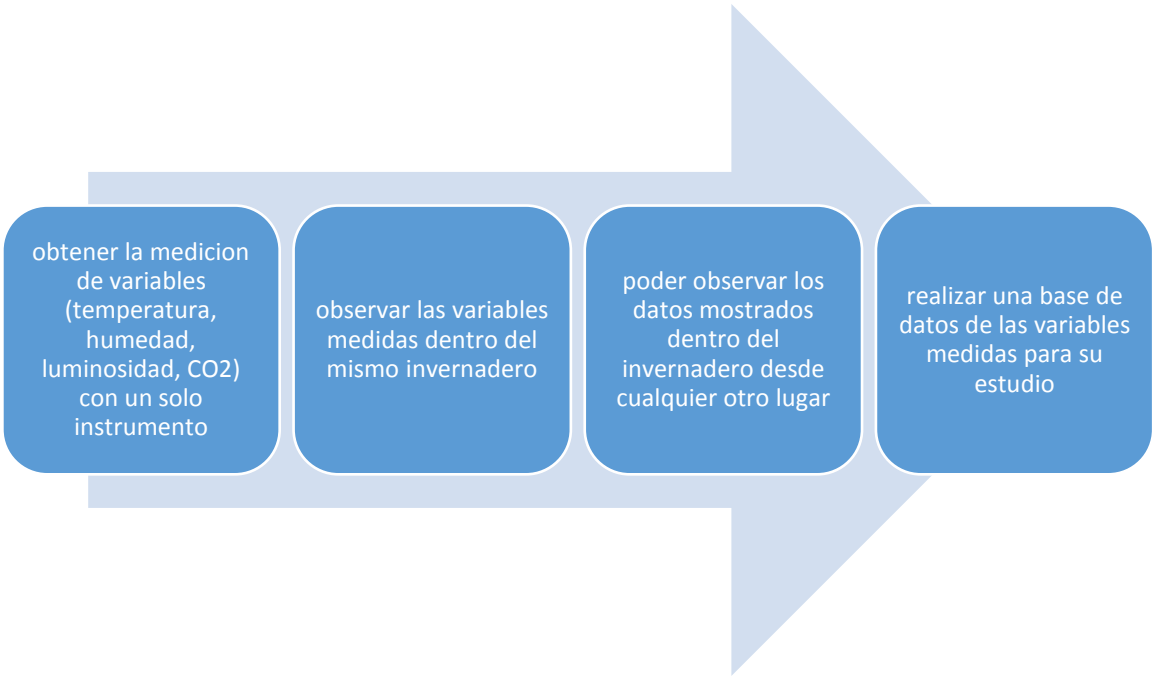
1.7.2 LIMITACIONES:

Una de las limitaciones que tiene este proyecto es que estará instalado en solo un invernadero de los 4 con los que cuenta el instituto, además que como su nombre lo dice, realizará solo monitoreo, la parte de corrección la harán los doctores, en cuanto ellos vean que las condiciones del ambiente en el invernadero no son las óptimas para el desarrollo del cultivo, ellos acudirán a realizar las correcciones al invernadero ya que la parte de control automático estaría en desarrollo en un futuro, además que los mismos doctores decidieron que no era necesario aun el control automático para dichas correcciones.

1.8 METODOLOGÍA PARA EL DESARROLLO DEL PROYECTO

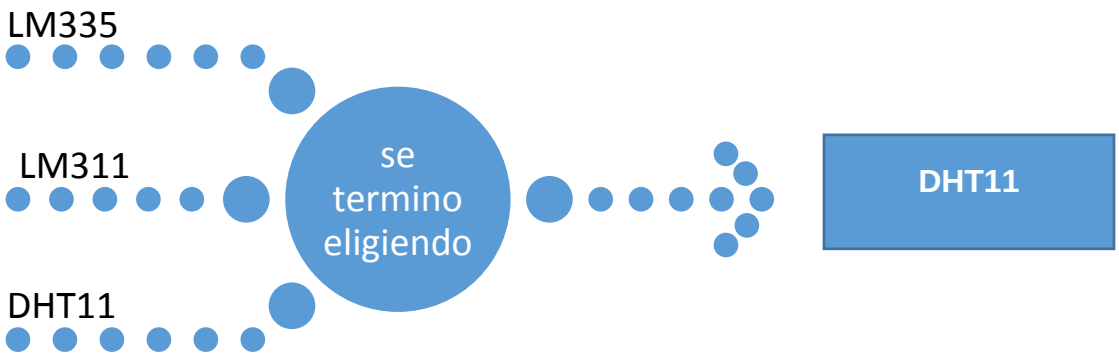
Par a poder monitorear un invernadero se necesita de varias partes que trabajen en conjunto sincronizadamente ya que de esta manera se puede entregar un resultado o se puede llegar a un objetivo o meta en sí. Por estar en el área de ingeniería electrónica, se buscó dar solución al problema implementando conocimiento adquirido durante la carrera, como es programación y manejo de hardware, se pudo aplicar el conocimiento adquirido para diseñar placas electrónicas pero por cuestiones de tiempo se decidió utilizar placas ya diseñadas para trabajar arduamente en programar e interconectar cada dispositivo.

Se planeó realizar un proyecto que realizara todas las tareas necesarias para satisfacer las necesidades de los interesados, los pasos planeados con los mismos fueron los siguientes:



1. Medir temperatura

   En el mercado existen variedades de sensores de temperatura por lo que hay muchos que se podían escoger, así que se hicieron las propuestas que más satisfacían la necesidad basado en precio, disponibilidad y respuesta del mismo.



2. Medir humedad en el ambiente

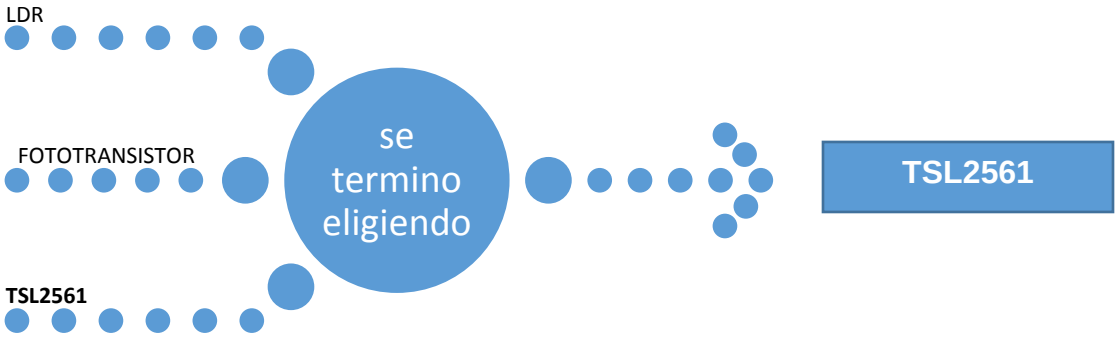
Para medir la humedad en el ambiente se decidió utilizar el mismo modelo de sensor DHT11 ya que este también tiene la opción de medir la humedad ambiental, configurándolo de diferente manera a nivel código.

3. Medir humedad de suelo

Para tener una lectura más rica de información se decidió añadir a las variables medidas la humedad del suelo, en este no hubo discusión alguna para la elección del sensor ya que se optó por utilizar el más conocido para usar con Arduino.

4. Medir luminosidad

La luminosidad dentro de un invernadero es muy importante ya que este influye en la fotosíntesis de las plantas, para obtener la medición de esta variable si se discutió sobre qué tipo de sensor se utilizaría, los tres principales fueron:



5. Medir CO2

El CO2 también es muy importante para las plantas por lo que el saber que tanto CO2 hay en el invernadero le ayudará bastante en la toma de decisiones a los encargados del invernadero. Se investigaron los diferentes tipos pero se llegó a la conclusión que era mejor utilizar uno diseñado por los mismos fabricantes de Arduino, por cuestiones de compatibilidad y fácil uso.

CAPÍTULO 2

FUNDAMENTOS TEÓRICOS

2.1 ESTACIÓN METEOROLÓGICA

Una estación meteorológica es un lugar escogido adecuadamente para colocar los diferentes instrumentos que permiten medir las distintas variables que afectan al estado de la atmosfera. Es decir es un lugar que nos permite la observación de los fenómenos atmosféricos y donde hay aparatos que miden las variables atmosféricas. Muchos de estos han de estar al aire libre, pero otros, aunque también han de estar al aire libre, deben estar protegidos de las radiaciones solares para que estas no les alteren los datos, el aire debe circular por dicho interior. Los que han de estar protegidos de las inclemencias del tiempo, se encuentran dentro de una garita meteorológica.

Una garita meteorológica es una casilla donde se instalan los aparatos del observatorio meteorológico que se deben proteger. Ha de ser una especie de casilla elevada un metro y medio del suelo (como mínimo elevada 120 cm) y con paredes en forma de persiana; éstas han de estar colocadas de manera que priven la entrada de los rayos solares en el interior para que no se altere la temperatura y la humedad. La puerta de la garita ha de estar orientada al norte y la teja debe estar ligeramente inclinada. En su interior están los instrumentos que han de estar protegidos como he dicho antes por aparatos registradores.

¿Cómo funciona?

La mayor parte de la estación meteorológica están automatizadas (E.M.A) requiriendo un mantenimiento ocasional. Existen observatorios meteorológicos sinópticos, que cuentan con personal (observadores), de forma que además de los datos anteriormente señalados se pueden recoger aquellos relativos a nubes, visibilidad y tiempo presente y pasado. La recogida de estos datos se denomina observaciones sinópticas.
Para la medida de variables en mares y océanos se utilizan sistemas dispuestos en boyas meteorológicas.
Otras instalaciones meteorológicas menos comunes disponen de instrumental de sondeo remoto como radar meteorológico para medir la turbulencia atmosférica y la actividad de tormentas. Estas y otras variables pueden obtenerse mediante el uso de globos sonda. [1]

2.2 VARIABLES MEDIDAS

2.2.1 Temperatura

La temperatura es una magnitud referida a las nociones comunes de caliente, tibio o frío que puede ser medida con un termómetro. En física, se define como una magnitud escalar relacionada con la energía interna de un sistema termodinámico, definida por el principio cero de la termodinámica. Más específicamente, está relacionada directamente con la parte de la energía interna conocida como «energía cinética», que es la energía asociada a los movimientos de las partículas del sistema, sea en un sentido traslacional, rotacional, o en forma de vibraciones. A medida de que sea mayor la energía cinética de un sistema, se observa que éste se encuentra más «caliente»; es decir, que su temperatura es mayor.
En el caso de un sólido, los movimientos en cuestión resultan ser las vibraciones de las partículas en sus sitios dentro del sólido.

En el caso de un gas ideal monoatómico se trata de los movimientos traslacionales de sus partículas (para los gases multiatómicos los movimientos rotacional y vibracional deben tomarse en cuenta también).
El desarrollo de técnicas para la medición de la temperatura ha pasado por un largo proceso histórico, ya que es necesario darle un valor numérico a una idea intuitiva como es lo frío o lo caliente.

Multitud de propiedades fisicoquímicas de los materiales o las sustancias varían en función de la temperatura a la que se encuentren, como por ejemplo su estado (sólido, líquido, gaseoso, plasma), su volumen, la solubilidad, la presión de vapor, su color o la conductividad eléctrica. Así mismo es uno de los factores que influyen en la velocidad a la que tienen lugar las reacciones químicas. La temperatura se mide con termómetros, los cuales pueden ser calibrados de acuerdo a una multitud de escalas que dan lugar a unidades de medición de la temperatura. En el Sistema Internacional de Unidades, la unidad de temperatura es el kelvin (K), y la escala correspondiente es la escala Kelvin o escala absoluta, que asocia el valor «cero kelvin» (0 K) al «cero absoluto», y se gradúa con un tamaño de grado igual al del grado Celsius. Sin embargo, fuera del ámbito científico el uso de otras escalas de temperatura es común. La escala más extendida es la escala Celsius, llamada «centígrada»; y, en mucha menor medida, y prácticamente sólo en los Estados Unidos, la escala Fahrenheit. También se usa a veces la escala Rankine (°R) que establece su punto de referencia en el mismo punto de la escala Kelvin, el cero absoluto, pero con un tamaño de grado igual al de la Fahrenheit, y es usada únicamente en Estados Unidos, y sólo en algunos campos de la ingeniería.

Temperatura ambiente

Temperatura ambiente es la temperatura que se puede medir con un termómetro y que se toma del ambiente actual, por lo que, si se toma de varios puntos en un área a un mismo tiempo puede variar.

Esto es debido a que una temperatura tomada en un ambiente tan frío como lo es el Polo Norte, donde la temperatura sería bajo cero (si se mide en grados Fahrenheit o en Centígrados), no será igual a una tomada en un lugar tan cálido como un desierto donde la temperatura estaría muy por encima del cero.
Uso científico
Para cálculos científicos, la temperatura ambiente es usualmente tomada como 20 ó 25 grados Celsius (293 ó 298 Kelvin, 68 ó 77 grados Fahrenheit). Por conveniencia numérica, 300.00 K (26.85 °C, 80.33 °F) es utilizado ocasionalmente, sin ser especificada como "temperatura ambiente". Sin embargo, la temperatura ambiente no es un término científico uniformemente definido, a diferencia de la Temperatura y Presión Estándar, o TPE, que tiene definiciones ligeramente diferentes.

Sensación térmica

La temperatura también se define como el grado de calor o frío que hay en un lugar, y podemos interpretar el frío como un sitio de baja temperatura y el calor, lo inverso.

2.2.2 Humedad

El aire de la atmósfera se considera normalmente como una mezcla de dos componentes: aire seco y agua. El agua es la única sustancia de la atmósfera que puede condensar (pasar de vapor a líquido) o evaporarse (pasar de líquido a vapor) en las condiciones ambientales que conocemos en la Tierra. Este hecho justifica la división del aire atmosférico en aire seco y agua, y además provocan una gran cantidad de fenómenos meteorológicos como la lluvia, el rocío, las nubes, etc. Además de todo esto, el estudio del agua en el aire atmosférico es esencial para la sensación de bienestar.
La capacidad atmosférica para recibir vapor de agua se relaciona con la humedad atmosférica, que corresponde a la cantidad de vapor de agua presente en el aire, originada por la evaporación del vital elemento desde los océanos, lagos y ríos. Se relaciona directamente con la temperatura, ya que las masas de aire cálido contienen mayor humedad que las de aire frío.

La humedad puede provocar diversas variaciones durante el día y entre un lugar a otro. Existe una cantidad límite de humedad que puede contener una masa de aire, denominada punto de saturación. Una vez traspasado ese umbral, el vapor

de agua contenido cambia de estado, se condensa y se convierte en precipitaciones. Estas últimas pueden presentarse como lluvia, granizo o nieve.

¿Cuándo hay saturación? Cuando el aire húmedo tiene una composición tal que está en equilibrio con una superficie libre plana de agua pura que tenga la misma temperatura que el aire. La palabra equilibrio implica que no hay, en total, transferencia de moléculas de vapor del aire a la superficie del agua, ni de la superficie del agua al aire. Cabe aclarar que las condiciones son diferentes en el caso de una superficie no plana o agua no pura (como en el caso de las gotas de nube, cuya superficie es curva y el agua que la forma tiene sustancias disueltas).
La humedad atmosférica se puede expresar de forma absoluta o de forma relativa como humedad relativa o grado de humedad.

¿Cómo se Mide?

El índice de temperatura – humedad (índice T – H, también llamado índice de incomodidad) expresa con un valor numérico la relación entre la temperatura y la humedad como medida de la comodidad o de la incomodidad. Se calcula sumando 40 al 72% de la suma de las temperaturas en un termómetro seco y en otro húmedo. Por ejemplo, si la temperatura en el termómetro seco es de 30ºC y en el húmedo es de 20ºC, el índice T – H será de 76. Cuando el valor es 70, la mayoría de la gente está cómoda, si el índice es de 75 el ambiente se hace más incómodo.

Clasificación

Humedad Absoluta: es la masa total de agua existente en el aire por unidad de volumen, y se expresa en gramos por metro cúbico de aire. La humedad atmosférica terrestre presenta grandes fluctuaciones temporales y espaciales.
Humedad Específica: se mide la masa de agua que se encuentra en estado gaseoso en un kilogramo de aire húmedo, y se expresa en gramos por kilogramo de aire.

Humedad Relativa: es la relación porcentual entre la cantidad de vapor de agua real que existe en la atmósfera y la máxima que podría contener a idéntica temperatura; es decir, es el cociente en la humedad absoluta y la cantidad máxima de agua que admite el aire por unidad de volumen. Se mide en tantos por ciento y está normalizada de forma que la humedad relativa máxima posible es el 100%. Una humedad relativa del 100% significa un ambiente en el que no cabe más agua. El cuerpo humano no puede transpirar y la sensación de calor puede llegar a ser asfixiante. Corresponde a un ambiente húmedo. Una humedad del 0% corresponde a un ambiente seco. Se transpira con facilidad.

Cuando la humedad alcanza el valor del 100% se produce fenómenos de condensación que observamos en la vida diaria. El fenómeno del rocío en las mañanas de invierno se debe a que la humedad relativa del aire ha alcanzado el 100% y el aire no admite ya más agua. Entonces el agua condensa en forma líquida en superficie metálicas, hojas, flores etc. También se alcanza el 100% de

humedad cuando usamos agua muy caliente en un recinto cerrado como por ejemplo un cuarto de baño. El agua caliente se evapora fácilmente y el aire de la habitación alcanza con rapidez el 100% de humedad. El resultado es de todos conocidos... se empañan (se humedecen) los espejos del baño.
Estos dos fenómenos son diferentes pero ilustran las dos formas en que puede aumentar la humedad de un recinto:

- Por disminución de la temperatura ambiental.
- Por aumento de la cantidad de agua en el ambiente

El primero de los fenómenos se relaciona con el concepto de temperatura de rocío. Si se mantiene la cantidad de agua del ambiente constante y se disminuye la temperatura llega un momento en que se alcanza una humedad relativa del 100%. Es momento es el punto de rocío y su temperatura la temperatura de rocío. Esto es justamente lo que ocurre en las madrugadas de invierno. La temperatura desciende tanto que llega al punto de rocío, en ese momento la humedad relativa del 100% hace que el agua se condense en las superficies.
Cualquier objeto de una habitación que tenga una temperatura menor que la temperatura de rocío presenta condensación en sus paredes por este fenómeno. Así ocurre por ejemplo cuando sacamos una lata de refresco de un frigorífico y la situamos en una mesa. Su temperatura es, seguramente, menor que la de rocío y observamos como la lata se empaña de humedad.
Los que usan gafas conocen perfectamente qué ocurre cuando, en una fría mañana de invierno, se introducen súbitamente en un recinto cerrado y caliente (por ejemplo en un autobús). La temperatura de los cristales de las gafas es muy baja y menor que la temperatura de rocío del recinto. Los cristales se empañan rápidamente hasta que se calientan y se sitúan a la temperatura del recinto.

2.2.3 Dióxido de carbono (CO2)

El dióxido de carbono es un gas incoloro e inodoro que se forma en todos aquellos procesos en que tiene lugar la combustión de sustancias que contienen carbono.
En ambientes interiores no industriales sus principales focos son la respiración humana y el fumar; aunque los niveles de dióxido de carbono también pueden incrementarse por la existencia de otras combustiones (cocinas y calefacción) o por la proximidad de vías de tráfico, garajes o determinadas industrias.

La concentración de dióxido de carbono en un ambiente exterior puede aportar información sobre distintos aspectos y circunstancias de un lugar tales como posibilidad de efectos sobre la salud de las personas cercanas al lugar, correlación con problemas y quejas por olor o como dato para estudiar el sitio en donde se encuentra algún rastro de concentración de dióxido de carbono.

EMISIONES DE CO2

La capa más baja de la Tierra llamada la tropósfera, existe un balance cíclico de gases que protegen y hacen posible la vida en el planeta. Entre los gases que contiene se encuentra el dióxido de carbono (CO2), un gas que contribuye a moderar la pérdida de calor de la Tierra al espacio exterior. Tanto el CO2, como el Metano, el óxido nitroso y el ozono, son conocidos como "gases de efectos invernadero", ya que su función es similar a las paredes de cristal de un invernadero; permiten que penetre la radiación solar en la atmósfera terrestre, pero evitan que escape, ya que tienen la propiedad de absorber parte de la radiación solar que recibe y refleja la superficie terrestre y conservarla como energía calorífica.
La mayoría de los gases de efecto Invernadero se generan naturalmente. En particular, el CO2 liberado a la atmósfera por procesos de descomposición orgánica y su concentración en la atmósfera es moderado por el crecimiento de las plantas.

Sin embargo, a partir de la Revolución Industrial, el ser humano comenzó a emitir grandes cantidades de CO2 por la quema de combustibles fósiles (carbón, petróleo y gas natural) y la quema de biomasa, que se han acumulado en la atmósfera.
Al haber una mayor absorción de radiación infrarroja, el aumento del CO2 atmosférico, junto con otros gases invernadero, se ha traducido en un aumento en la temperatura terrestre; fenómeno conocido como Cambio Climático.

FUENTES DE CO2

Entre las fuentes que emiten los gases y partículas contaminantes a la atmósfera, se tiene: industrias, fábricas, plantas de producción de energía, vehículo, hogares, entre otros.

Estas fuentes pueden ser puntuales como una chimenea o afectar a una región a través de una serie de fuentes pequeñas. Las fuentes de contaminación se pueden clasificar en: móviles y estacionarias.

2.2.4 Rayos ultravioleta

Definición de radiación UV
La radiación que recibe la tierra proviene del sol, esta radiación comprende una gama continua y muy extensa de longitudes de onda que van desde los rayos gama a las ondas de radio, pasando por los rayos X, ultravioleta (UV), visible infrarrojo (IR) y microondas.
La radiación ultravioleta (UV) cuya longitud de onda comprendida aproximadamente entre los 400nm y los 100nm tiene una longitud de onda más corta que la luz visible. (Figura 2.1).

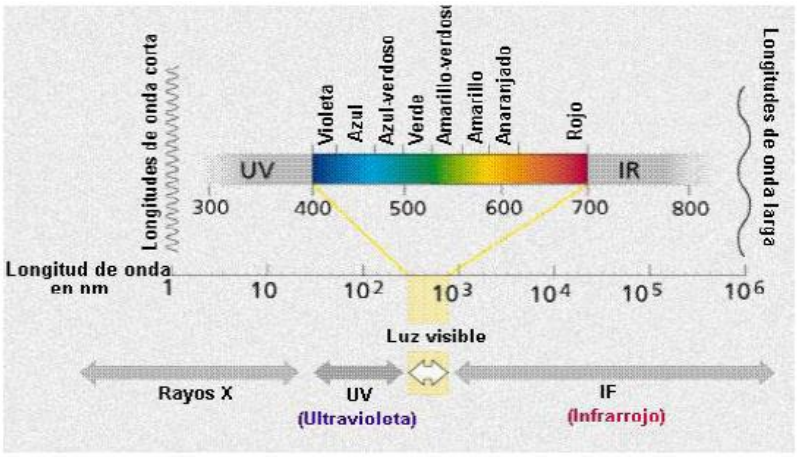


Figura 2.1: Espectro de radiación solar

La radiación de longitud de onda entre 100 y 300nm se le conoce como ultravioleta lejano o de vacío. Comúnmente proviene del sol o de lámparas de descarga gaseosa. El oxígeno y el nitrógeno de la atmosfera absorben la radiación ultravioleta lejana proveniente del sol, transformando su energía en reacciones fotoquímicas e impidiendo, en consecuencia, que llegue a la superficie terrestre, donde imposibilitara a la existencia de la vida. [4]

Clasificación de radiación UV

Los rayos solares, conocidos como también como los ultravioleta (UV) son invisibles al ojo humano y se clasifica de acuerdo a su longitud de onda, que se medida en nanómetros (nm). Cabe destacar que entre más corta es la onda, más intensa es la energía de los rayos solares.

Este espectro se puede subdividir en tres zonas (figura 2.2):

- La radiación tipo UV-A; que se comprende la radiación solar menos nociva. La longitud de onda se encuentra entre los 320 y 400 nm y la mayoría de estos rayos llegan a la superficie terrestre.
- La radiación tipo UV-B; o los rayos de media onda; la longitud de esta onda se encuentra entre los 290 y 320 nm, esta energía en gran parte es absorbida por la capa de ozono antes de llegar a la superficie terrestre.
- La radiación tipo UV-C cuya longitud de onda oscila entre los 200 y 290 nm. Estos rayos son absorbidos por la capa de ozono antes de llegar a la tierra y son potencialmente peligrosos para los seres humanos.

Dicho de una manera específica, cuando la luz solar atraviesa la atmosfera, el ozono, el vapor de agua, el oxígeno y el dióxido de carbono absorben toda la radiación UV-C y aproximadamente el 70% de la radiación UV-B.

La atmosfera absorbe la radiación UV-A en menor medida, entonces, se puede decir que la radiación UV que alcanza la superficie terrestre se compone en su mayor parte de rayos UV-A con pequeña porción de rayos UV-B [5]

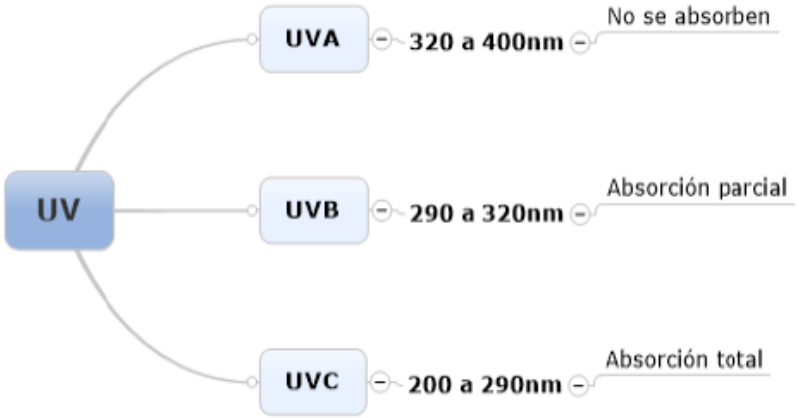


Figura 2.2: Clasificación de la radiación UV

Efectos positivos y negativos

El sol es imprescindible para la vida y produce efectos benéficos para nuestro organismo, pero una exposición sin protección y de una forma desmedida le ocasionará efectos nocivos. En este caso, además, al tratarse de una exposición a radiación ultravioleta, es importante tener en cuenta algunos factores más, tales como la elevación solar, atenuación de la radiación, nubosidad y albedo, que son factores que influyen en cuanta radiación UV incide sobre la superficie terrestre.

2.3 Instrumentos de medición (sensores):

2.3.1 DHT11: Sensor de humedad/temperatura para Arduino.

El DHT11 es un sensor de temperatura muy económico y muy utilizado para la electrónica y los microcontroladores como Arduino. El DHT11 es un sensor que proporciona una salida de datos digital. Entre sus ventajas podemos mencionar el bajo costo y el despliegue de datos digitales. Esto supone una gran ventaja frente a los sensores del tipo análogo, como el LM335 por ejemplo, en los cuales las fluctuaciones en el voltaje alteran la lectura de datos.

Entre las desventajas pues, el DHT11 solo lee enteros, no podemos leer temperaturas con decimales por lo que tenemos que pensarlo muy bien a la hora de utilizar este sensor para trabajos en los que se requieran lecturas precisas de temperatura y/o humedad.

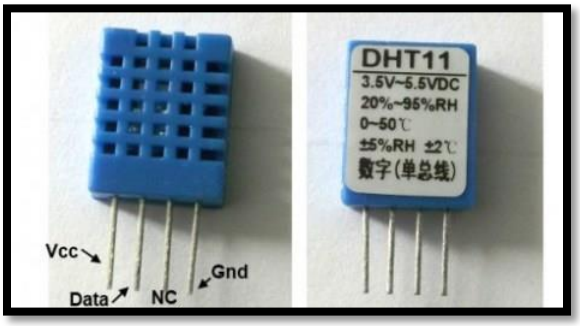


Figura 2.1 sensor DHT11

2.3.2 MQ-2 Sensor de CO2.

La serie de sensores de gas MQ utiliza un pequeño calentador de interior con un sensor electroquímico. Son sensibles para una gama de gases y se utilizan en interiores a temperatura ambiente. La salida es una señal analógica y se puede leer con una entrada analógica del Arduino. El módulo MQ-2 Sensor de Gas es útil para la detección de fugas de gas en el hogar y la industria. Es capaz de detectar LPG, i-butano, propano, metano, alcohol, CO2, el hidrógeno y el humo. Algunos módulos tienen una resistencia variable incorporada para ajustar la sensibilidad del sensor.

Figura 2.2 sensor de CO2 MQ-2

2.3.3 Sensor de luz TSL2561 Adafruit para Arduino.

El TSL2561 es un sensor de luz digital avanzado, ideal para usar en un amplio rango de aplicaciones. Permite realizar cálculos de lux exactos, incorporando dos sensores: infrarrojo y de luz visible, de manera de realizar mediciones de ambos espectros por separado o en conjunto. El consumo de corriente es muy bajo, lo que permite el uso en sistemas de registro alimentados por baterías.

Características:

El sensor de luz visible aproxima la respuesta del ojo humano

Medición exacta de iluminancia en diversas condiciones de iluminación

Rango dinámico: 0.1 a 40000 lux.

Rango de temperaturas de operación: -30 a +80 ºC.

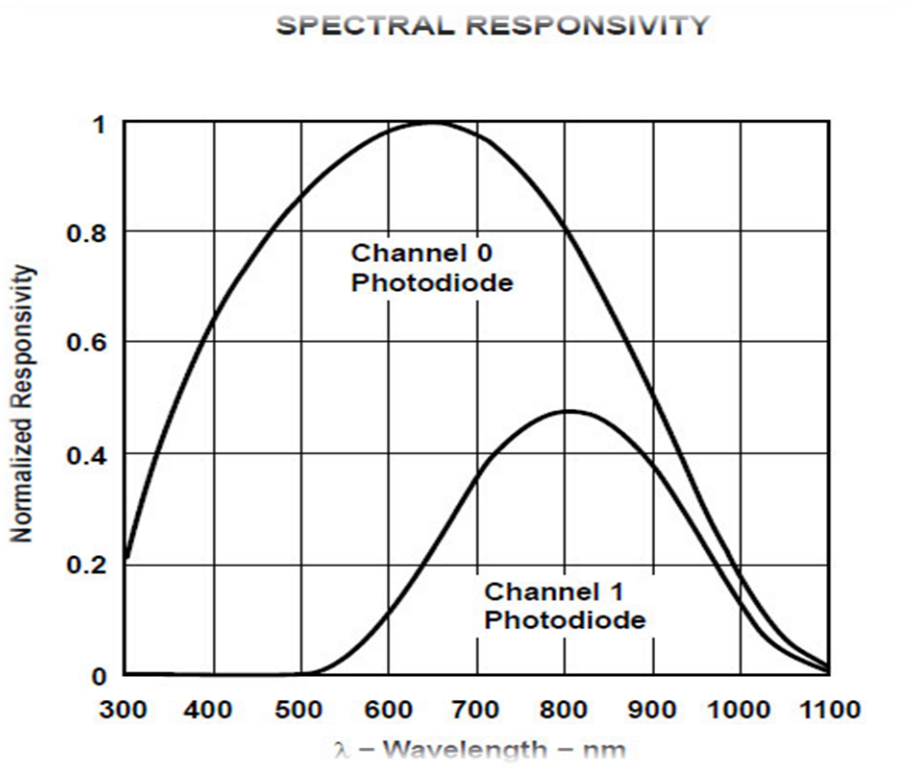
Rango de tensiones de alimentación: 2.7 a 3.6 VDC.

Interface: I2C.

Figura 2.3 sensor de luminosidad TSL2561 Adafruit

Comportamiento del sensor de luz.

Tabla 2.1. Comportamiento del sensor de luminosidad TSL2561



Algunas estadísticas

- Se aproxima respuesta del ojo humano
- Mide Precisamente iluminancia en condiciones de iluminación Diversos
- Rango de temperatura: -30 a 80 * C
- El rango dinámico (Lux): 0,1 a 40.000 Lux
- Rango de tensión: 2.7-3.6V
- Interfaz: I2C

2.3.4 Sensor PIR (Rev B).

El sensor PIR detecta el movimiento mediante la medición de los cambios en los niveles de infrarrojos (calor) emitidos por los objetos circundantes. Cuando se detecta el movimiento del sensor PIR emite una señal de alto en su pin de salida.

- Mayor rango de detección, seleccionable por puente a bordo.
- Gran tensión de alimentación, de 3 a 6 VDC.
- Corriente de salida superior, se ofrece para el control directo de una carga externa.
- Orificios de montaje incluidos para proyectos permanentes.
- Todas las piezas de SMT.

Características principales:

- Normalmente detecta una persona hasta aproximadamente 30 pies de distancia, o hasta 15 pies de distancia en el modo de sensibilidad reducida.
- Un Jumper selecciona el funcionamiento normal o sensibilidad reducida.
- Aproximadamente 90 grados de campo de vista.
- LED bordo ilumina la lente para la retroalimentación visual rápido cuando se detecta movimiento.
- Orificios de montaje para tornillos de tamaño proporcionan una fácil integración en aplicaciones permanentes.
- Paquete SIP 3 pines perfecto para proyectos de breadboard amigable.
- Fácil comunicación con cualquier microcontrolador.
- Tamaño pequeño hace que sea fácil de ocultar.

Figura 2.4 Sensor PIR (Rev B).

Ideas de aplicación:

Luz de noche Motion-activted.

Sistema de alarmas.

Apoyos vacaciones animadas.

2.3.5 sensor de humedad de suelo SEE-SEN92355

Este sensor viene con los 2 conductores que clavaremos en una maceta o jardinera y trae 3 pines GND, VCC y DAT. Al ser un sensor analógico sólo necesita alimentación y una entrada de datos analógica.

Tabla 2.2 Pines del sensor de humedad [SEE-SEN92355]

Sensor	Arduino
VCC	5V
GND	GND
DAT	A0

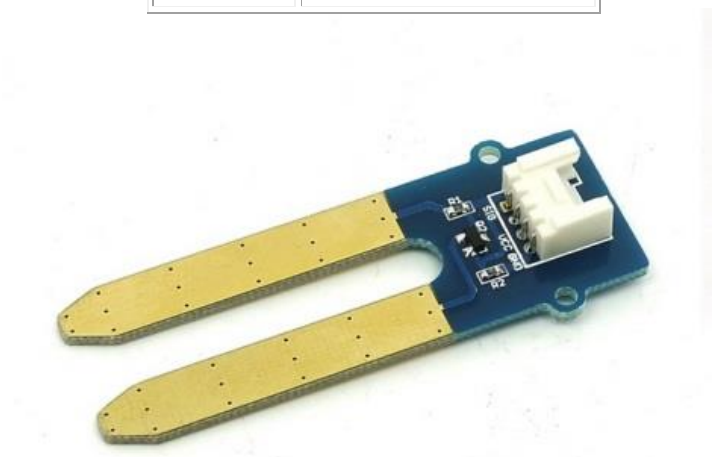


Figura 2.5 sensor de humedad de suelo SEE-SEN92355

2.4 Instrumentos de medición (microcontroladores):

2.4.1 Arduino Mega 2560.

El Nuevo Arduino Mega 2560 (rev3) viene con más memoria para el programa, más RAM y más pines. 100% compatible con la versión UNO o Duemilanove. En ésta novedosa versión, se utiliza un potente procesador de AVR ATMEGA2560 con un amplio espacio de memoria para programar y corriendo a 16Mhz. Es ideal para proyectos de robótica ya que lo más destacado es su elevada cantidad de pines de entrada y salida y sus 4 puertos UART por hardware.

Las placas Shield de la versión Duemilanove / UNO siguen siendo compatibles con éste modelo ya que los pines básicos estás situados en el mismo sitio, por lo que podremos seguir utilizando cualquier placa shield del modelo Duemilanove. La única diferencia es que la placa Mega es ligeramente más alargada, por lo que podremos acceder a una nueva hilera de pines con los pines IO adicionales.

Ésta placa es una revolución para el mundo de la robótica o los proyectos con grandes necesidades de memoria para el programa.

Características:

- Microcontrolador: ATmega2560
- Tensión de alimentación: 5V
- Tensión de entrada recomendada: 7-12V
- Límite de entrada: 6-20V
- Pines digitales: 54 (14 con PWM)
- Entradas analógicas: 16
- Corriente máxima por pin: 40 mA
- Corriente máxima para el pin 3.3V: 50 mA
- Memoria flash: 256 KB
- SRAM: 8 KB
- EEPROM: 4 KB
- Velocidad de reloj: 16 MHz
- Documentación:
- Datasheet ATMega2560
- Descripción oficial en arduino.cc
- Descargar del compilador

Figura 2.6 Arduino Mega 2560

2.4.2 Arduino Yun.

El Arduino Yun es el primer miembro de una nueva serie de placas Arduino que combinan la potencia de Linux junto con la sencillez característica de Arduino. Combina el chip del modelo Leonardo (ATMega32U4) junto con un módulo SOC (System-On-a-Chip) corriente de una distribución de Linux llamada Linino, basada en OpenWRT. Una de las características más interesantes es que soporta red cableada Ethernet y Wifi. El chip Arduino está conectado al módulo Linux, por lo que es muy fácil que se comuniquen entre ambos y delegar procesos pesados a la máquina Linux integrada en la placa.

Conectividad:

Dispone de dos conexiones de red. Una red Ethernet 10/100 Mbps y otra Wifi (IEEE 802.11 b/g/n, 2,4GHz) que puede montarse como cliente o como punto de acceso.

Conexión entre procesadores:

Para comunicar el pequeño ATMega32U4 con el módulo Linux, se utiliza la librería Bridge, que facilita mucho las cosas y es soportada directamente por el team de Arduino. El puerto serial del AR9331 está conectado al serial del 32U4 con los pines 0 y 1. El puerto serie del AR9331 es un acceso a la consola CLI, lo que permite lanzar procesos y recuperar mensajes directamente desde la consola.

Varios paquetes de gestión del sistema de archivos y administración ya están preinstalados por defecto (incluso el intérprete de Python) y la librería bridge permite también instalar y lanzar aplicaciones propias con ese mismo sistema. Una de las ventajas más interesantes, es que la placa (del lado del 32U4) puede ser programada directamente por Wifi a través del módulo Linux. Es una placa llena de posibilidades. También cabe destacar que dispone de un zócalo para memoria MicroSD que permite almacenar datos en ella como páginas web, datos o cualquier otra cosa que necesitemos, ampliando aún más las posibilidades de la placa.

Características:

Las características del lado del Arduino son iguales a su hermano pequeño Arduino Leonardo. La parte Linux tienes las siguientes especificaciones:

- Procesador: Atheros AR9331
- Arquitectura: MIPS @400MHz
- Alimentación: 3.3V
- Puerto Ethernet: IEEE 802.3 10/100Mbit/s
- Conexión Wifi: IEEE 802.11b/g/n
- USB Type-A: 2.0 Host/Device
- Lector de tarjetas: Micro-SD
- RAM: 64 MB DDR2
- Memoria Flash: 32 MB
- Soporte para PoE tipo 802.3af

Figura 2.7 Arduino Yun

2.5 Interfaces

2.5.1 Display LCD 20x4 Arduino

Una pantalla de cristal líquido (LCD) es una pantalla de panel plana, representación visual electrónica, o pantalla de vídeo que utiliza la luz modulación propiedades de los cristales líquidos. Los cristales líquidos no emiten luz directamente.

Aquí, en este proyecto vamos a utilizar un monocromática LCD alfanumérico 20x4.20x4 significa que 20 caracteres se pueden mostrar en cada una de las 4 filas de la pantalla LCD 20x4, así un total de 80 caracteres se puede visualizar en cualquier instancia de tiempo.

Figura 2.8 Display LCD 20x4 Arduino

2.5.2 Logitech HD Webcam C270

Especificaciones:

- Videoconferencias HD (1280 x 720 píxeles) con el sistema recomendado
- Captura de vídeo: Hasta 1280 x 720 píxeles
- Tecnología Logitech Fluid Crystal™
- Fotos: Hasta 3.0 megapíxeles (mejora por software)
- Micrófono integrado con reducción de ruido
- Certificación USB 2.0 de alta velocidad (se recomienda)
- Clip universal para monitores LCD, CRT o portátiles

Software de cámara Web Logitech:

- Controles de panorámico, inclinación y zoom
- Captura de vídeo y fotos
- Seguimiento facial
- Detección de movimiento

Figura 2.9 Logitech HD Webcam C270

2.5.3 Router

Un router también conocido como enrutador o encaminador de paquetes es un dispositivo que proporciona conectividad a nivel de red o nivel tres en el modelo OSI. Su función principal consiste en enviar o encaminar paquetes de datos de una red a otra, es decir, interconectar subredes, entendiendo por subred un conjunto de máquinas IP que se pueden comunicar sin la intervención de un encaminador (mediante puentes de red), y que por tanto tienen prefijos de red distintos.

Funcionamiento

El funcionamiento básico de un enrutador o encaminador, como se deduce de su nombre, consiste en enviar los paquetes de red por el camino o ruta más adecuada en cada momento. Para ello almacena los paquetes recibidos y procesa la información de origen y destino que poseen. Con arreglo a esta información reenvía los paquetes a otro encaminador o bien al anfitrión final, en una actividad que se denomina 'encaminamiento'. Cada encaminador se encarga de decidir el siguiente salto en función de su tabla de reenvío o tabla de encaminamiento, la cual se genera mediante protocolos que deciden cuál es el camino más adecuado o corto, como protocolos basado en el algoritmo de Dijkstra.

Por ser los elementos que forman la capa de red, tienen que encargarse de cumplir las dos tareas principales asignadas a la misma:

Reenvío de paquetes: cuando un paquete llega al enlace de entrada de un encaminador, éste tiene que pasar el paquete al enlace de salida apropiado. Una característica importante de los encaminadores es que no difunden tráfico difusivo.

Encaminamiento de paquetes: mediante el uso de algoritmos de encaminamiento tiene que ser capaz de determinar la ruta que deben seguir los paquetes a medida que fluyen de un emisor a un receptor.

Por tanto, debemos distinguir entre reenvío y encaminamiento. Reenvío consiste en coger un paquete en la entrada y enviarlo por la salida que indica la tabla, mientras que por encaminamiento se entiende el proceso de hacer esa tabla.

Tipos de enrutamiento

Tanto los enrutadores como los anfitriones guardan una tabla de enrutamiento. El daemon de enrutamiento de cada sistema actualiza la tabla con todas las rutas conocidas. El núcleo del sistema lee la tabla de enrutamiento antes de reenviar paquetes a la red local. La tabla de enrutamiento enumera las direcciones IP de las redes que conoce el sistema, incluida la red local predeterminada del sistema. La tabla también enumera la dirección IP de un sistema de portal para cada red conocida. El portal es un sistema que puede recibir paquetes de salida y reenviarlos un salto más allá de la red local.

Enrutamiento estático

Hosts y redes de tamaño reducido que obtienen las rutas de un enrutador predeterminado, y enrutadores predeterminados que sólo necesitan conocer uno o dos enrutadores

Determinación de enrutamiento:

La información de enrutamiento que el encaminador aprende desde sus fuentes de enrutamiento se coloca en su propia tabla de enrutamiento. El encaminador se vale de esta tabla para determinar los puertos de salida que debe utilizar para retransmitir un paquete hasta su destino. La tabla de enrutamiento es la fuente principal de información del enrutador acerca de las redes. Si la red de destino está conectada directamente, el enrutador ya sabrá el puerto que debe usar para reenviar los paquetes. Si las redes de destino no están conectadas directamente, el encaminador debe aprender y calcular la ruta más óptima a usar para reenviar paquetes a dichas redes. La tabla de enrutamiento se constituye mediante uno de estos dos métodos o ambos:

- Manualmente, por el administrador de la red.
- A través de procesos dinámicos que se ejecutan en la red.

Rutas estáticas

Las rutas estáticas se definen administrativamente y establecen rutas específicas que han de seguir los paquetes para pasar de un puerto de origen hasta un puerto de destino. Se establece un control preciso de enrutamiento según los parámetros del administrador.

Las rutas estáticas por defecto especifican una puerta de enlace de último recurso, a la que el enrutador debe enviar un paquete destinado a una red que no aparece en su tabla de enrutamiento, es decir, se desconoce.

Las rutas estáticas se utilizan habitualmente en enrutamientos desde una red hasta una red de conexión única, ya que no existe más que una ruta de entrada y salida en una red de conexión única, evitando de este modo la sobrecarga de tráfico que genera un protocolo de enrutamiento. La ruta estática se configura para conseguir conectividad con un enlace de datos que no esté directamente conectado al enrutador. Para conectividad de extremo a extremo, es necesario configurar la ruta en ambas direcciones. Las rutas estáticas permiten la construcción manual de la tabla de enrutamiento.

Enrutamiento dinámico

El enrutamiento dinámico le permite a los encaminadores ajustar, en tiempo real, los caminos utilizados para transmitir paquetes IP. Cada protocolo posee sus propios métodos para definir rutas (camino más corto, utilizar rutas publicadas por pares, etc.).

Introducción a RIP

RIP (Protocolo de Información de Enrutamiento) es uno de los protocolos de enrutamiento más antiguos utilizados por dispositivos basados en IP. Su implementación original fue para el protocolo Xerox a principios de los 80. Ganó popularidad cuando se distribuyó con UNIX como protocolo de enrutamiento para esa implementación TCP/IP. RIP es un protocolo de vector de distancia que utiliza la cuenta de saltos de enrutamiento como métrica. La cuenta máxima de saltos de RIP es 15. Cualquier ruta que exceda de los 15 saltos se etiqueta como inalcanzable al establecerse la cuenta de saltos en 16. En RIP la información de enrutamiento se propaga de un enrutador a los otros vecinos por medio de una difusión de IP usando protocolo UDP y el puerto 520.

Figura 2.10 ROUTER TPLINK GENÉRICO

2.6 RECOPILACIÓN DE DATOS

2.6.1 Base de datos

Se le llama base de datos a los bancos de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto.

Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido; una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Actualmente, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital, siendo este un componente electrónico, por tanto se ha desarrollado y se ofrece un amplio rango de soluciones al problema del almacenamiento de datos.

Existen programas denominados sistemas gestores de bases de datos, abreviado SGBD (del inglés Database Management System o DBMS), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos DBMS, así como su utilización y administración, se estudian dentro del ámbito de la informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas; También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

Aunque las bases de datos pueden contener muchos tipos de datos, algunos de ellos se encuentran mutuamente protegidos por las leyes de varios países.

Clasificación de bases de datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al contexto que se esté manejando, la utilidad de las mismas o las necesidades que satisfagan según la variabilidad de la base de datos.

Bases de datos estáticas

Son bases de datos únicamente de lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones, tomar decisiones y realizar análisis de datos para inteligencia empresarial.

Bases de datos dinámicas

Son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado y edición de datos, además de las operaciones fundamentales de consulta. Un ejemplo, puede ser la base de datos utilizada en un sistema de información de un supermercado.

Según el contenido

Bases de datos bibliográficas

Sólo contienen un subrogante (representante) de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque si no, estaríamos en presencia de una base de datos a texto completo (o de fuentes primarias —ver más abajo). Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

Bases de datos de texto completo[editar]

Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.

CAPÍTULO 3

DESARROLLO E IMPLEMENTACIÓN DEL PROYECTO

3.1 CARACTERÍSTICAS DEL SISTEMA DE MONITOREO

Se utilizó una tarjeta Arduino Mega, la cual se encarga de recibir la información dada por sensores de humedad, temperatura, CO2, y luminosidad, todos estos sensores están colocados estratégicamente dentro del invernadero para realizar una medición general del invernadero, esta información es recogida por el Arduino Mega, éste procesa la información y la envía al Arduino Yun el cual cuenta con una interfaz Ethernet, este procesa toda la información y la envía por medio de un modem a una red LAN, toda la información transmitida también es monitoreada por un software el cual va haciendo una base de datos de dicha información.

Toda la información obtenida de la red podemos observarla solo en la computadora que esté conectada al mismo modem, por lo que otra máquina fuera del "área" o que no esté conectada al modem, no puede ver la información, o puede ver la información también conectándose directamente a la red WIFI del Arduino Yun, pero para poder subir esa información a internet necesitamos el modem, ya que será el Router interno que trae el modem el que dará la ruta para poder observar la información desde la web.

En el Arduino Yun están conectados un sensor de movimiento y una cámara web, el sensor al detectar movimiento activa la cámara la cual toma una imagen del lugar, esa imagen es guardada en una carpeta de Dropbox a la cual se puede acceder desde la web desde el lugar que este con solo tener una cuenta. La página de YouTube cuenta con una herramienta para usuarios, la cual permite transmitir video en vivo, por lo que se decidió utilizarla. En el diagrama 1 se puede observar un diagrama a bloques de la forma de operación del sistema.

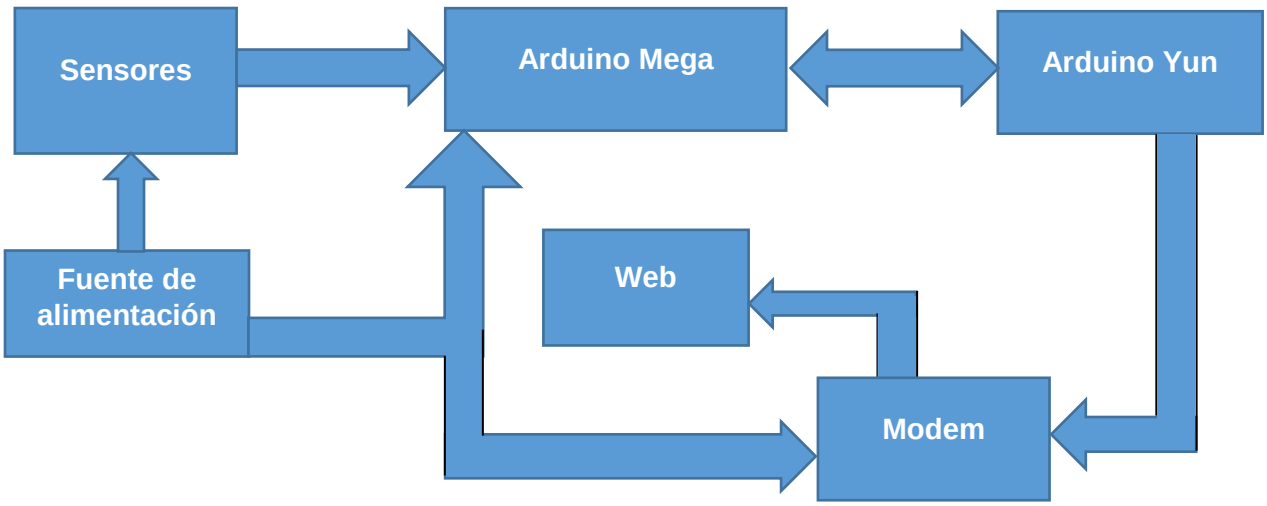


Diagrama 1. Funcionamiento del sistema.

El dispositivo es compacto y apto para realizar las mediciones en el sitio que se requiera. Cuenta en su interior con una placa Arduino Mega, la cual hace el proceso de lectura de los sensores y codifica la información. También cuenta con una placa Arduino Yun montada en la misma placa Arduino Mega, la placa Arduino Yun se encarga de recibir la información procesada por el Arduino Mega y la envía al modem.

En el mismo prototipo se visualiza la información obtenida por los sensores en una pantalla de cristal líquido, mostrando la medición de temperatura, humedad, luminosidad, CO2 y humedad de suelo.

En el mismo interior cuenta con un ventilador pequeño pero eficaz que mantiene la temperatura normalizada para el correcto funcionamiento de las placas Arduino.

La elección de los sensores se basó en viabilidad, disponibilidad y costos de los mismos

3.2 SENSORES UTILIZADOS

3.2.1 Arduino – sensor de humedad

Lo primero es configurar los sensores con las placas Arduino, un ejemplo de cómo conectar estos sensores es de la siguiente manera, se empezó con el sensor DTH11 guiado de acuerdo al diagrama 2.

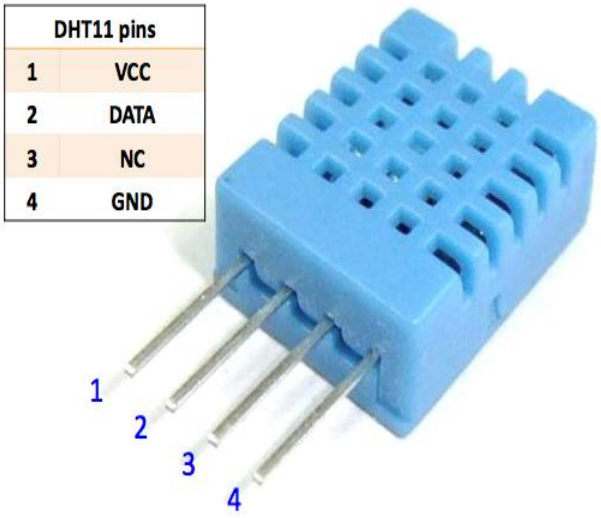


Diagrama 2. Pines de conexiones DHT11

Una vez que ya se sabe cómo conectar este sensor a la placa, se procede a hacer la configuración en nuestra placa, teniendo cuidado de conectar bien. En la figura 3.1 se observa un ejemplo de cómo conectar el sensor a la placa Arduino. Estos ejemplos son proporcionados en la página del fabricante y foros de Arduino, por lo que se procedió a conectar de la misma manera solo variando el pin al que se le entrega la información y declarándolo en el código.

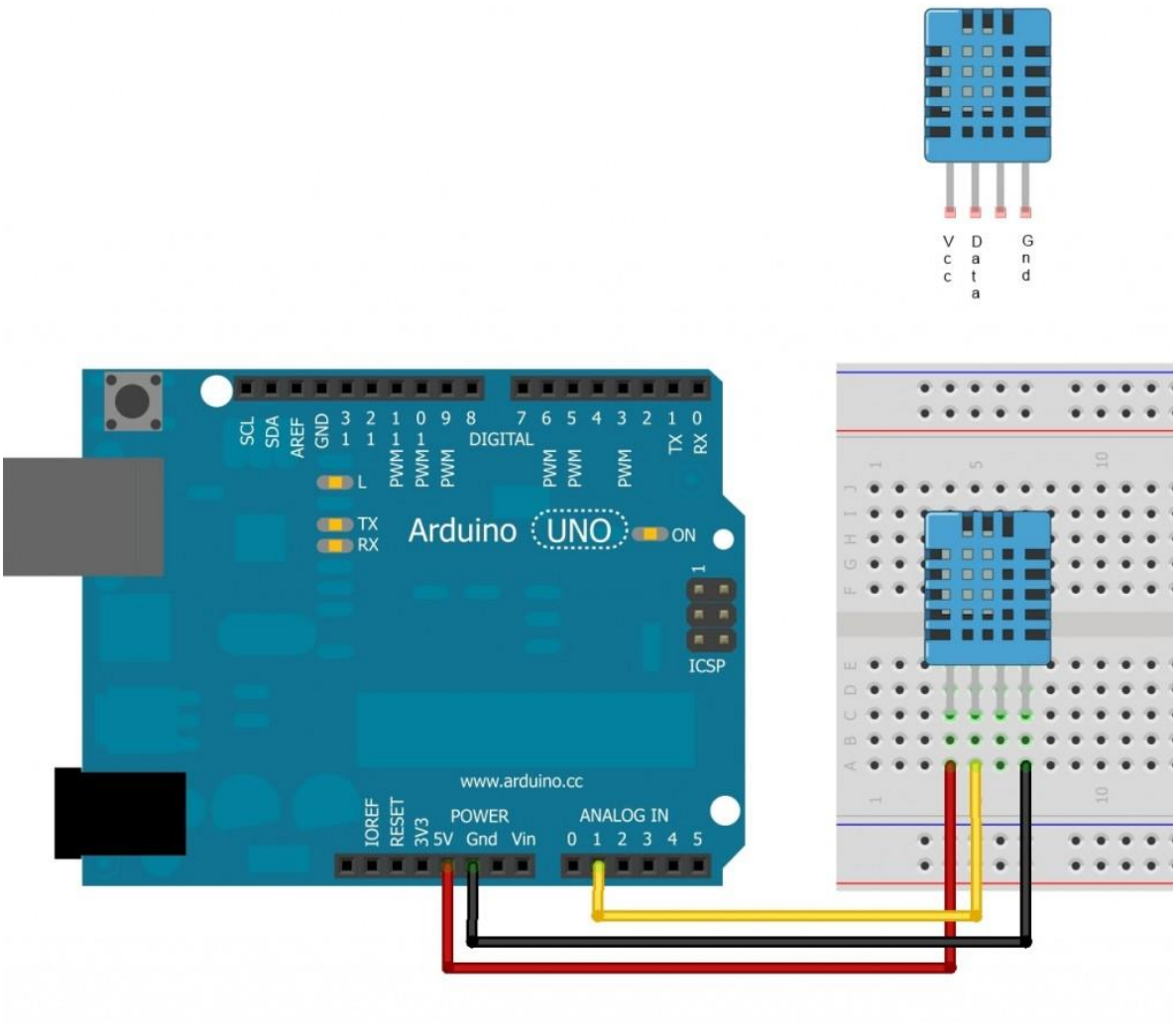


Figura 3.1 conexión sensor DHT11 con Arduino

Ya una vez conectado el sensor con el Arduino se procedió a probarlo verificando que este funcione correctamente. Para hacer esto nos guiamos de este programa de ejemplo para hacer dicha prueba:

Programa de ejemplo para probar el sensor. Comprueba que tu sensor funciona correctamente con DHT11Lib. Este código muestra en pantalla la información del sensor. Se requiere copiar la siguiente carpeta en “libraríes”: link.

```
/*

#define DHT11PIN 2#include <dht11.h>

dht11 DHT11;

void setup()
{
Serial.begin(9600);
Serial.println(“DHT11 TEST PROGRAM “);
Serial.print(“LIBRARY VERSION: “);
Serial.println(DHT11LIB_VERSION);
}

void loop()
{
Serial.println(“\n”);

int chk = DHT11.read(DHT11PIN);

Serial.print(“Read sensor: “);
switch (chk)
{
case 0: Serial.println(“OK”); break;
case -1: Serial.println(“Checksum error”); break;
```

```
case -2: Serial.println("Time out error"); break;
default: Serial.println("Unknown error"); break;
}

Serial.print("Humidity (%): ");
Serial.println((float)DHT11.humidity, DEC);

Serial.print("Temperature ( °C): ");
Serial.println((float)DHT11.temperature, DEC);

Serial.print("Temperature ( °F): ");
Serial.println(DHT11.fahrenheit(), DEC);

Serial.print("Temperature ( °K): ");
Serial.println(DHT11.kelvin(), DEC);

Serial.print("Dew Point ( °C): ");
Serial.println(DHT11.dewPoint(), DEC);

Serial.print("Dew PointFast ( °C): ");
Serial.println(DHT11.dewPointFast(), DEC);

delay(2000);
}
```

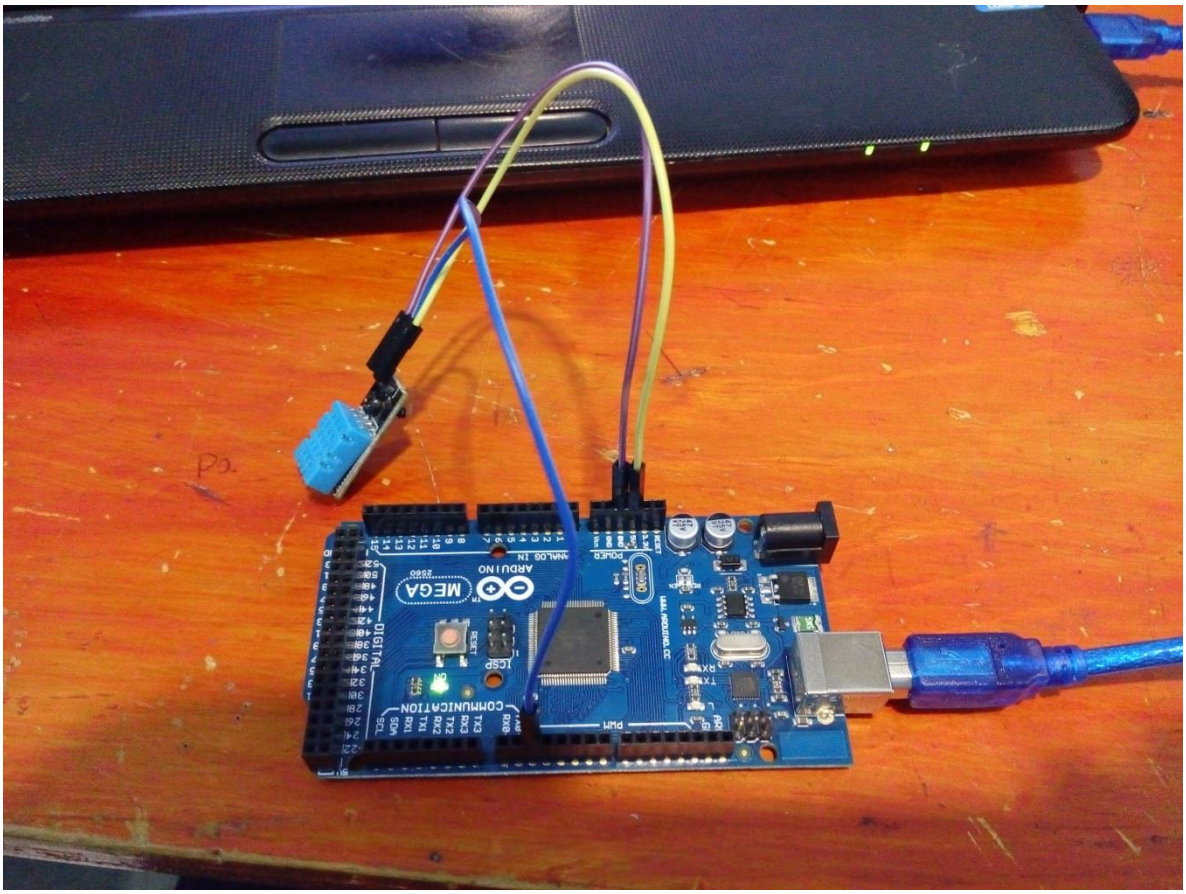


Figura 3.2 conectando el sensor DHT11 con el Arduino

3.2.2 Arduino – sensor de humedad de suelo

Ahora procedemos a configurar el sensor de humedad de suelo con la placa Arduino, para poder conectar este sensor debemos guiarnos de sus especificaciones técnicas tal y como se muestra en el diagrama 3.

Tabla 3.1. Especificaciones de conexión del sensor de humedad de suelo

Cable rojo	+5V
Cable negro	GND
Cable amarillo	Salida de señal analógica

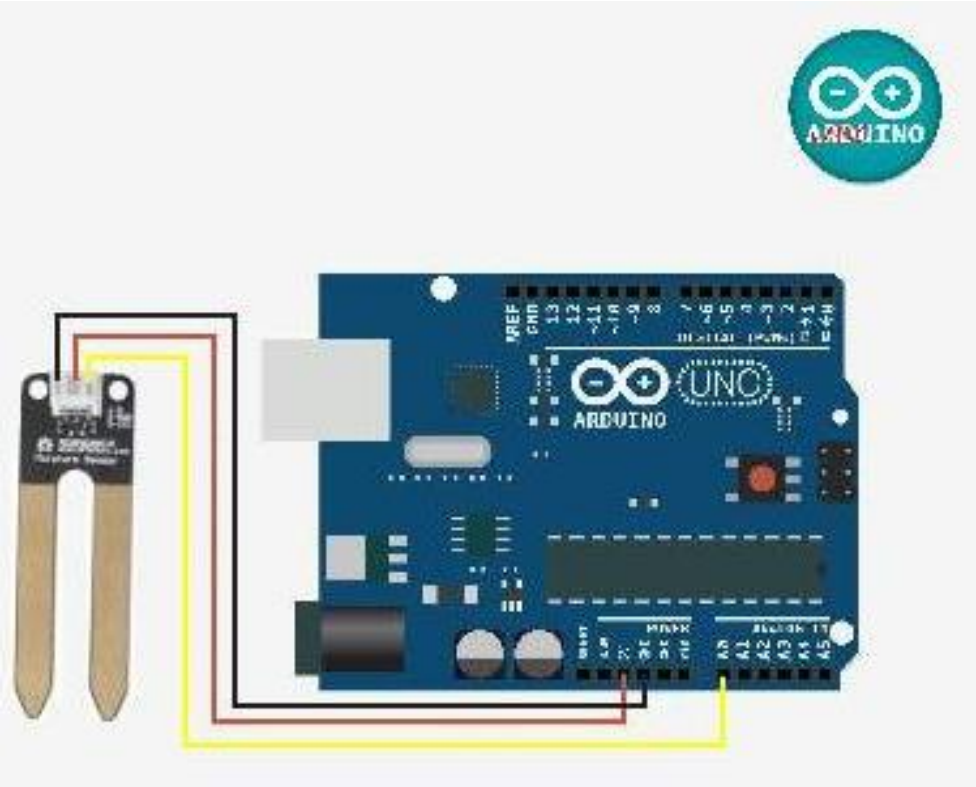


Diagrama 3. Conexión sensor de humedad de suelo - Arduino

Configuramos el sensor de luminosidad, para eso probamos primero el sensor para saber que este funcione correctamente, para ello se utilizó este ejemplo para la prueba.

```
// Sensor de Humedad de Suelo

// Conectamos el sensor de la siguiente forma:
// GND    -> GND
// VCC    -> 5V
// DAT    -> A0
// Conectamos a las entrada Analógica 0
  // Descripción de valores del Sensor

// 0  -300     Seco
```

```
// 300-700     Húmedo
// 700-950     En Agua

int Valor;

void setup(){
  Serial.begin(9600);
  Serial.println("http://arubia45.blogspot.com.es");
}

   void loop(){
      Serial.print("Sensor de Humedad valor:");
      Valor = analogRead(0);
      Serial.print(Valor);
      if (Valor <= 300)
        Serial.println(" Seco, necesitas regar");
      if ((Valor > 300) and (Valor <= 700))
        Serial.println(" Humedo, no regar");
      if (Valor > 700)
        Serial.println(" Encharcado");
      delay(1000);
   }
```

3.2.3 Arduino – sensor de luminosidad

Ahora procedemos a conectar el sensor de luminosidad con el Arduino, para ello nos guiamos de acuerdo a la conexión de ejemplo que se muestra en el figura 3.3

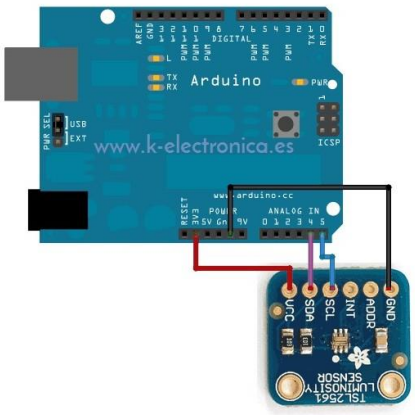


Figura 3.3 conexión Arduino – sensor de luminosidad

Ahora procedemos a probar el sensor para su correcto funcionamiento con el Arduino, para ello probaremos con el siguiente código, el cual prácticamente es el mismo usado para el proyecto solo con algunas modificaciones y ajustes.

- *// La dirección será diferente dependiendo de si usted deja*
- *// El flotador pin ADDR (0x39 addr), ni lo ate a tierra o VCC. En esos casos*
- *// Utilizar TSL2561_ADDR_LOW (0x29) o TSL2561_ADDR_HIGH (0x49), respectivamente*
- Adafruit_TSL2561 TSL = Adafruit_TSL2561 (TSL2561_ADDR_FLOAT, 12345);
- / ** ********************** /
- / *
- *Configura el tiempo de ganancia y de integración para el TSL2561*
- * /
- / ** ********************** /
- void configureSensor (vacío)

- {
- / * *También puede ajustar manualmente la ganancia o habilitar el soporte de auto-ganancia* * /
- // Tsl.setGain (TSL2561_GAIN_1X); / * Ninguna ganancia ... usar en la luz brillante para evitar la saturación del sensor * /
- // Tsl.setGain (TSL2561_GAIN_16X); / * 16x ganancia ... utilice con poca luz para aumentar la sensibilidad * /
- TSL. enableAutoRange (verdadera); / * *Auto-ganancia ... cambia automáticamente entre 1x y 16x* * /
-
- / * *Cambiar el tiempo de integración le da una mejor resolución del sensor* (402ms = datos de 16 bits) * /
- TSL. setIntegrationTime (TSL2561_INTEGRATIONTIME_13MS); / * *rápido pero de baja resolución* * /
- // Tsl.setIntegrationTime (TSL2561_INTEGRATIONTIME_101MS); / * *Resolución y la velocidad media* * /
- // Tsl.setIntegrationTime (TSL2561_INTEGRATIONTIME_402MS); / * *Datos de 16 bits, pero las conversiones más lentos* * /
-
- / * *Actualización de estos valores en función de lo que haya establecido más arriba!* * /
- Serial. Println ("------------------------------------");
- Serial. Print ("ganancia"); de serie. Println ("Auto");
- Serial. Print ("Tiempo:"); de serie. Println ("13 ms");
- Serial. Println ("------------------------------------");
- }
- / * *Obtener un nuevo evento de sensor* * /
- sensors_event_t evento;
- TSL. getEvent (y de eventos);
-
- / * *Mostrar los resultados (la luz se mide en lux)* * /
- si (evento. la luz)
- {
- Serial. De impresión (evento. La luz); de serie. Println ("lux");
- }
- más
- {
- / * Si event.light = 0 lux el sensor está probablemente saturado
- y no hay datos fiables podrían generarse! * /
- Serial. Println ("sobrecarga del sensor");

1. }uint16_t banda ancha = 0;
2. uint16_t infrarrojos = 0;
3.

4. */ * Llenar la banda ancha y de infrarrojos con los últimos valores * /*
5. getLuminosity (y de banda ancha, y de infrarrojos);
6. }
7. }

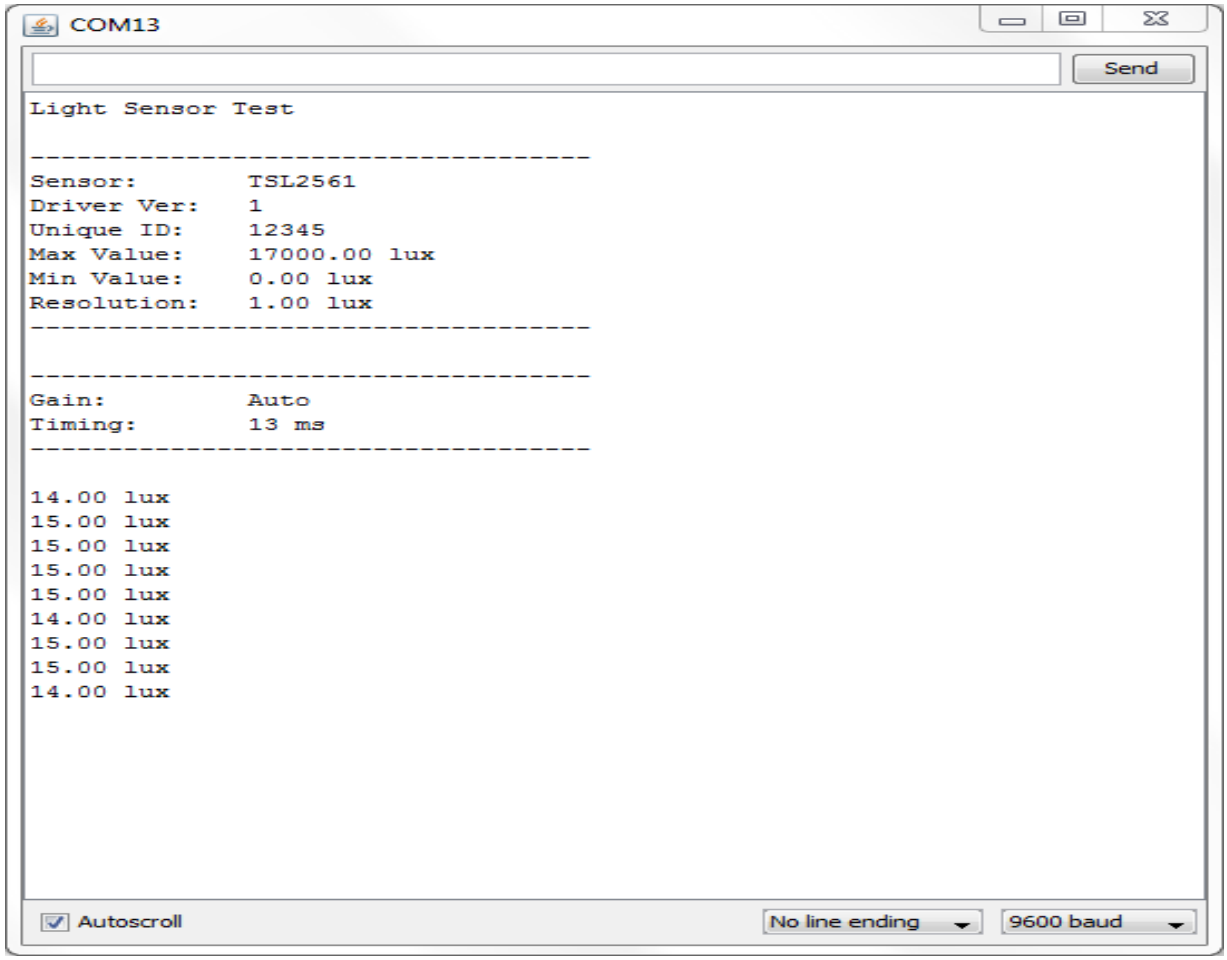


Figura 3.4 Sensor de luminosidad visto desde serial-monitor del Arduino.

3.1.4 Arduino – sensor de CO2

Configuramos el sensor de CO2, para ello nos guiamos de la figura 3.5 como ejemplo para conectar el sensor al Arduino.

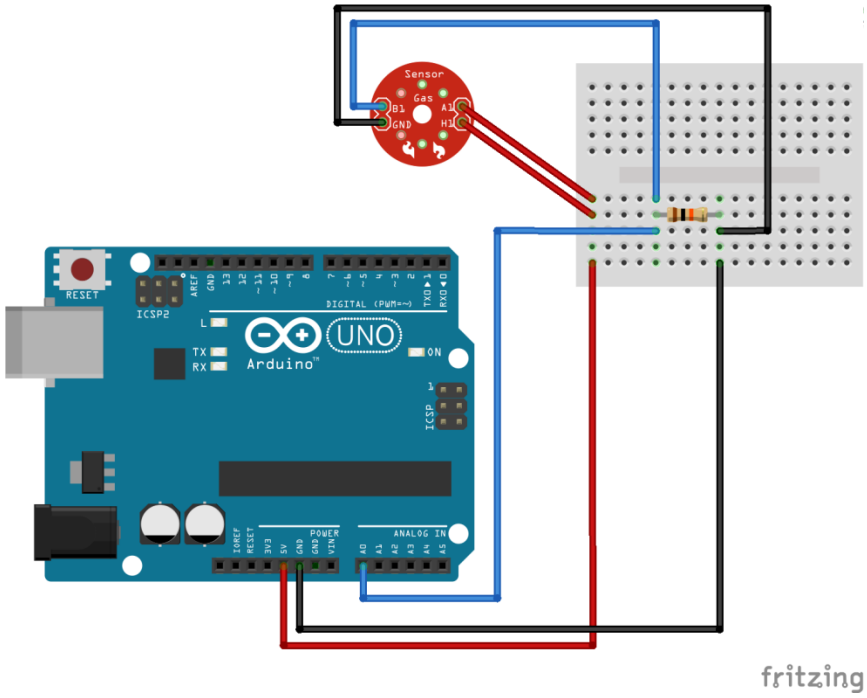


Figura 3.5 conexión Arduino – sensor de CO2

Ahora procedemos a configurar el sensor en la plataforma de Arduino, nos guiamos con los pasos proporcionados por el mismo fabricante para que el componente funcione correctamente.

```
/*
 # This Sample code is for display the CO2 sensor output on LCD12864.

 # Editor : Phoebe
 # Date   : 2014.1.15
 # Ver    : 0.1
 # Product: LCD12864 Shield for Arduino
 # SKU    : DFR0287

 # Hardwares:
 1. Arduino UNO
```

```
 2. LCD12864 Shield for Arduino
 3. CO2 Sensor


 #Steps:
 1.Connect the CO2 Sensor to the Analog pin 0
 2.Power the UNO board with 7~12V


*/


#include "U8glib.h"


U8GLIB_NHD_C12864 u8g(13, 11, 10, 9, 8);   // SPI Com: SCK = 13, MOSI = 11, CS = 10, A0 = 9, RST = 8


#define         MG_PIN                  (1)     //define which analog input channel you are going to use
#define         BOOL_PIN                (2)
#define         DC_GAIN                 (8.5)   //define the DC gain of amplifier


#define         READ_SAMPLE_INTERVAL        (50)    //define how many samples you are going to take in normal operation
#define         READ_SAMPLE_TIMES           (5)     //define the time interval(in milis econd) between each samples in


#define         ZERO_POINT_VOLTAGE          (0.324) //define the output of the sen sor in volts when the concentration of CO2 is 400PPM
#define         REACTION_VOLTGAE            (0.020) //define the voltage drop of the sensor when move the sensor from air into 1000ppm CO2


float           CO2Curve[3]  =  {
  2.602,ZERO_POINT_VOLTAGE,(REACTION_VOLTGAE/(2.602-3))};
```

```
int percentage;
float volts;

/*****************************  LCD12864 display ***********************************
Display the voltage & ppm data on the LCD12864
*********************************************************************************/
void draw() {
  u8g.setFont(u8g_font_unifont);
  u8g.drawStr( 0,11,"Voltage:");
  u8g.setPrintPos(70,11);
  u8g.print(volts);
  u8g.drawStr( 110, 11,"V" );
  u8g.drawStr(0,30,"CO2:");
  if (percentage == -1) {
    u8g.drawStr( 40,30,"<400" );
  }
  else {
    u8g.setPrintPos(40,30);
    u8g.print(percentage,DEC);
  }
  u8g.drawStr( 80,30,"ppm" );
  u8g.drawStr( 3, 63,"www.DFRobot.com" );
}

/*****************************  MGRead ***************************************
Input:   mg_pin - analog channel
Output:  output of CO2 Sensor
Remarks: This function reads the output of CO2 Sensor
*********************************************************************************/
float MGRead(int mg_pin)
```

```
{
  int i;
  float v=0;

  for (i=0;i<READ_SAMPLE_TIMES;i++) {
    v += analogRead(mg_pin);
    delay(READ_SAMPLE_INTERVAL);
  }
  v = (v/READ_SAMPLE_TIMES) *5/1024 ;

  return v;
}
/*****************************  MQGetPercentage **********************************
Input:   volts   - CO2 Sensor output measured in volts
         pcurve  - pointer to the curve of the target gas
Output:  ppm of the target gas
Remarks: By using the slope and a point of the line. The x(logarithmic value of ppm
)
         of the line could be derived if y(MG-811 output) is provided. As it is a
         logarithmic coordinate, power of 10 is used to convert the result to non-logarit
hmic
         value.
*****************************************************************************/
int  MGGetPercentage(float volts, float *pcurve)
{
  if ((volts/DC_GAIN )>=ZERO_POINT_VOLTAGE) {
    return -1;
  }
  else {
    return pow(10, ((volts/DC_GAIN)-pcurve[1])/pcurve[2]+pcurve[0]);
  }
```

```
}

void setup() {
  u8g.setRot180();// rotate screen, if required
}

void loop() {

  volts = MGRead(MG_PIN);
  percentage = MGGetPercentage(volts,CO2Curve);
  u8g.firstPage();
  do  {
    draw();
  }
  while( u8g.nextPage() );
  delay(200);
}
```

3.1.5 Arduino – sensor de movimiento

Configuramos el sensor de movimiento con el Arduino, para conectar el sensor a la placa nos guiamos del ejemplo mostrado en la figura 3.6

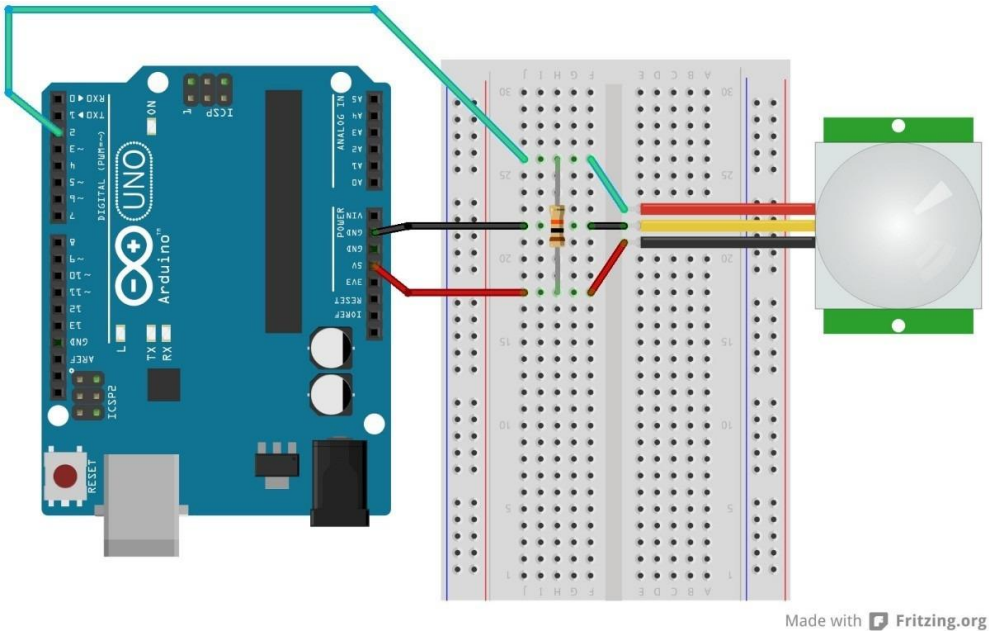


Figura 3.6 conexión Arduino – sensor de movimiento

Con el siguiente código procedemos a configurar el sensor con el Arduino:

```
//El tiempo que nos da el sensor a calibrar (10-60 segundos de acuerdo con la hoja de datos)
int calibrationTime = 10;

//El tiempo cuando el sensor emite un impulso de baja
long unsigned int lowIn;

//La cantidad de milisegundos el sensor tiene que ser bajo
//Antes de que asuma todo el movimiento se ha detenido
long unsigned int pause = 5000;

boolean lockLow = true;
boolean takeLowTime;

int pirPin = 7;    //pin digital conectado a la salida del sensor PIR
```

```
int ledPin = 8;



////////////////////////////
//SETUP
void setup(){
  Serial.begin(9600);
  pinMode(pirPin, INPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(pirPin, LOW);

  //Dar el sensor de un cierto tiempo para calibrar
  Serial.print(" calibrando sensor ");
    for(int i = 0; i < calibrationTime; i++){
      Serial.print(".");
      delay(1000);
      }
    Serial.println(" hecho");
    Serial.println("SENSOR ACTIVADO");
    delay(50);
  }

////////////////////////////
//LOOP
void loop(){

     if(digitalRead(pirPin) == HIGH){
       digitalWrite(ledPin, HIGH);   //LED visualiza el estado de la salida del sensor
       if(lockLow){
         //Se asegura de que esperar a que la transición a la baja antes de cualquier
salida se hace otra:
         lockLow = false;
         Serial.println("---");
         Serial.print("movimiento encontrado a los ");
         Serial.print(millis()/1000);
         Serial.println(" segundos ");
         delay(50);
```

```
        }
        takeLowTime = true;
      }

    if(digitalRead(pirPin) == LOW){
      digitalWrite(ledPin, LOW);  //LED visualiza el estado de la salida del sensor

      if(takeLowTime){
       lowIn = millis();          // guardar el momento de la transición de alta a baja
       takeLowTime = false;       //  asegurarse de que esto sólo se hace al comienzo
de una fase de baja
      }
      //Si el sensor es baja para más de la pausa dada,
      //Asumimos que el movimiento no más que va a pasar

      if(!lockLow && millis() - lowIn > pause){
          //Se asegura de este bloque de código sólo se ejecuta de nuevo después
de
          //Una secuencia de movimiento se ha detectado nueva
          lockLow = true;
          Serial.print("movimiento detenido a los ");
          Serial.print((millis() - pause)/1000);
          Serial.println(" segundos ");
          delay(50);
          }
      }
   }
```

3.3 Configuración de tarjetas Arduino e interfaz

3.3.1 Conexión del Display con Arduino

Se procedió a hacer la interfaz con el usuario, así que se conectó el display para que el usuario pueda ver las mediciones que le interesan sin tener que estar conectado a la computadora. Se guio de la figura 3.7 para hacer las conexiones correctas.

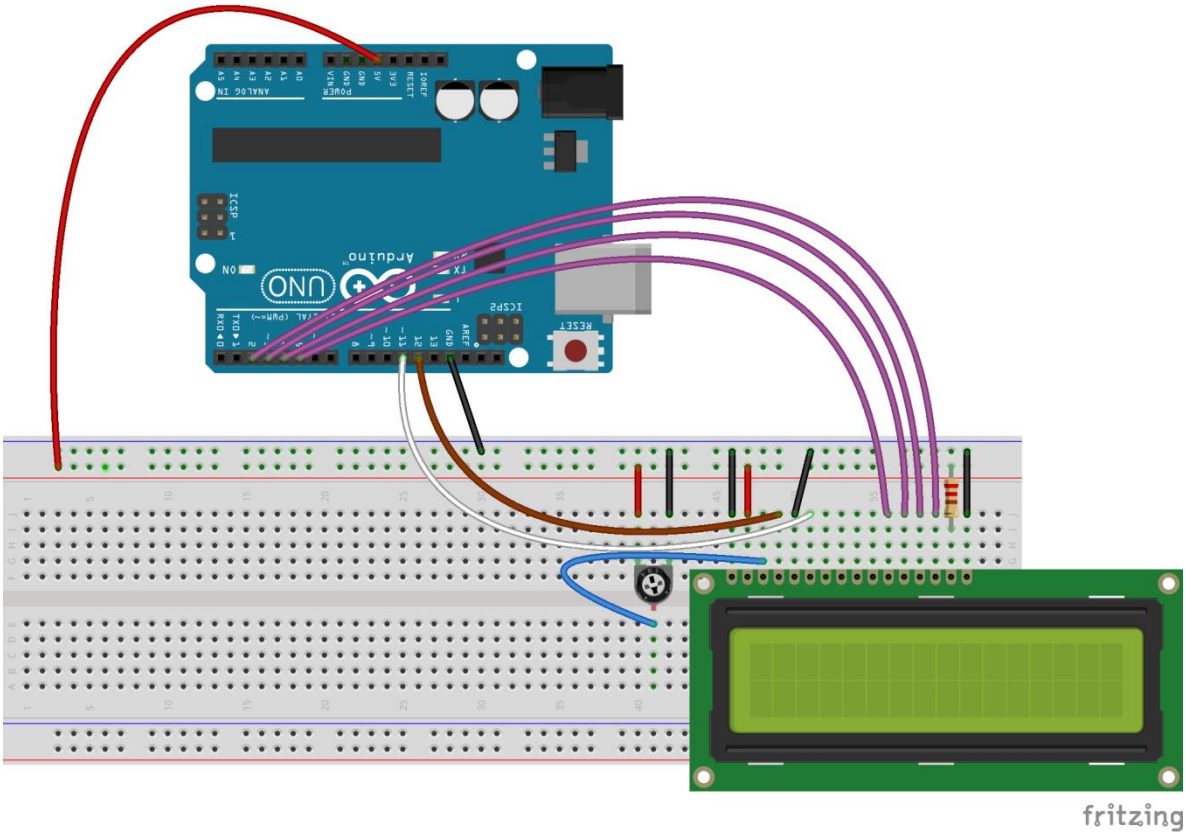


Figura 3.7 conexión del display con el Arduino

Configuración de la pantalla con Arduino en código por lo que se guio del código proporcionado por Arduino.cc [12]

```
//reallocates the printing area on the SC2004CSWB by Usmar A Padow
usmpadow@gmail.com and Greg Cox
#include<string.h>
#include <LiquidCrystal.h>
//LiquidCrystal lcd(12,11,2,3,4,5,6,7,8,9);       // my setup (matches original 4-bit
```

```
example)
LiquidCrystal lcd(A0, A1, 6, 7, 8, 9, A2, A3, A4, A5);
char buffer[20*4+1]="                                                                ";//20*4
spaces
int pos=0;
void render_realloc() {
      //clear screen
      lcd.clear();
      //print in correct order
      for(int pos2=0;pos2<20;pos2++)  lcd.print(buffer[pos2]);
      for(int pos2=40;pos2<60;pos2++) lcd.print(buffer[pos2]);
      for(int pos2=20;pos2<40;pos2++) lcd.print(buffer[pos2]);
      for(int pos2=60;pos2<80;pos2++) lcd.print(buffer[pos2]);
}
void print2(char str[]) {
      int pos2,pos3,strpo,b,spos2,posa,strpos,line;
      int len = strlen(str);
      //int overrun = pos+len-79;
      int overrun = pos+len-80;
      if(overrun>0) {
            if(len>=80) {//if buffer will fill the whole screen
                  //for(strpo=len-79, b=0;strpo<len;strpo++, b++) {
                  for(strpo=len-80, b=0;strpo<len;strpo++, b++) {//fill the screen with
the last 80 characters
                        buffer[b]=str[strpo];
                  }
                  //greg put thisbuffer[79]=' ';
                  //pos=79;
                  pos=80;
                  render_realloc();
                  return;
            }
            int charsleft = 80-pos;//calculate the ammount of chars that still fit in the
buffer
            for(posa = pos, strpos=0; strpos<=charsleft;posa++,strpos++) {//put the
rest of the characters into the buffer
                  buffer[posa]= str[strpos];
            }
            int remainder = overrun % 20;
            int extralines = 1+(overrun - remainder)/20; //divide by 20, removing
remainder (adding 1 because at least one line of overrun)
            for (line=0;line<extralines;line++) {
                  //scroll the display up by one line
```

```
                for(pos2=20;pos2<40;pos2++) buffer[pos2-20]=buffer[pos2];//put
line 2 on line 1
                for(pos2=40;pos2<60;pos2++) buffer[pos2-20]=buffer[pos2];//put
line 3 on line 2
                for(pos2=60;pos2<80;pos2++) buffer[pos2-20]=buffer[pos2];//put
line 4 on line 4
                for(pos2=60,pos3=0;pos2<80;pos2++,pos3++) {
                        int strp = charsleft+pos3+line*20;
                    char ch;
                    if (strp>=len) {
                            ch = ' ';
                    } else {
                            ch = str[strp];
                    }
                    buffer[pos2]=ch;
                }
                pos-=20;
            }
        } else {//if there is no overrun, just copu the string to the buffer
            for(spos2=0, pos2=pos;spos2<len;spos2++,pos2++) {
                buffer[pos2]=str[spos2];
            }
        }
        pos+=len;
        render_realloc();
}

void newline() {
        char spaces[21] = "                    ";// 20 spaces
        int remainder = pos % 20;//remember the 0-19 clock story
        int charsleft = 20-remainder;
        spaces[charsleft]= '\0';
        print2(spaces);
}

void print2ln(char str[]) {
        print2(str);
        newline();
}

void setup()  {
   lcd.begin(20, 4);
}
```

```
void cls() {
	lcd.clear();
	strcpy(buffer,"                                                                ");//20*4
spaces
	pos=0;
}
void loop()  {
	char tmp[100];
	cls();
	for (int i=47; i<127; i++)    {
		char tmp[2];
		sprintf(tmp,"%c",i);
		print2(tmp);
	}
	delay(5000);
	//char tmp[500];
	strcpy(tmp,"hello world!");
	print2(tmp);
	delay(5000);
	newline();
	newline();
	strcpy(tmp,"test");
	print2ln(tmp);
	delay(5000);
	strcpy(tmp,"1234567890abcdefghijklmnopqrstuvwxy");
	print2(tmp);
	delay(5000);
	strcpy(tmp,"x");
	print2(tmp);
	delay(5000);
	strcpy(tmp,"10 PRINT \"NO BEER FOR YOU!\"");
	print2(tmp);
	delay(5000);
}
```

3.3.2 Configuración del Arduino Yun para vincularlo con una red

PASO 1

Lo primero es conectar la placa por puerto USB:

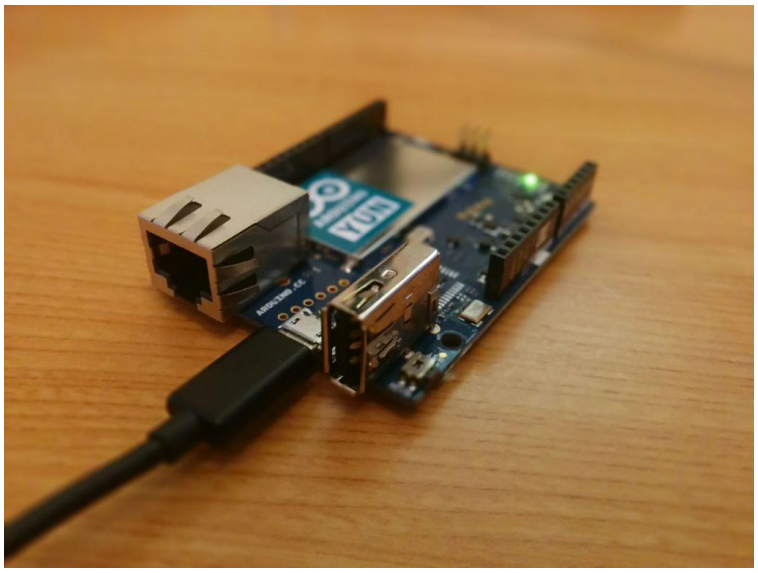
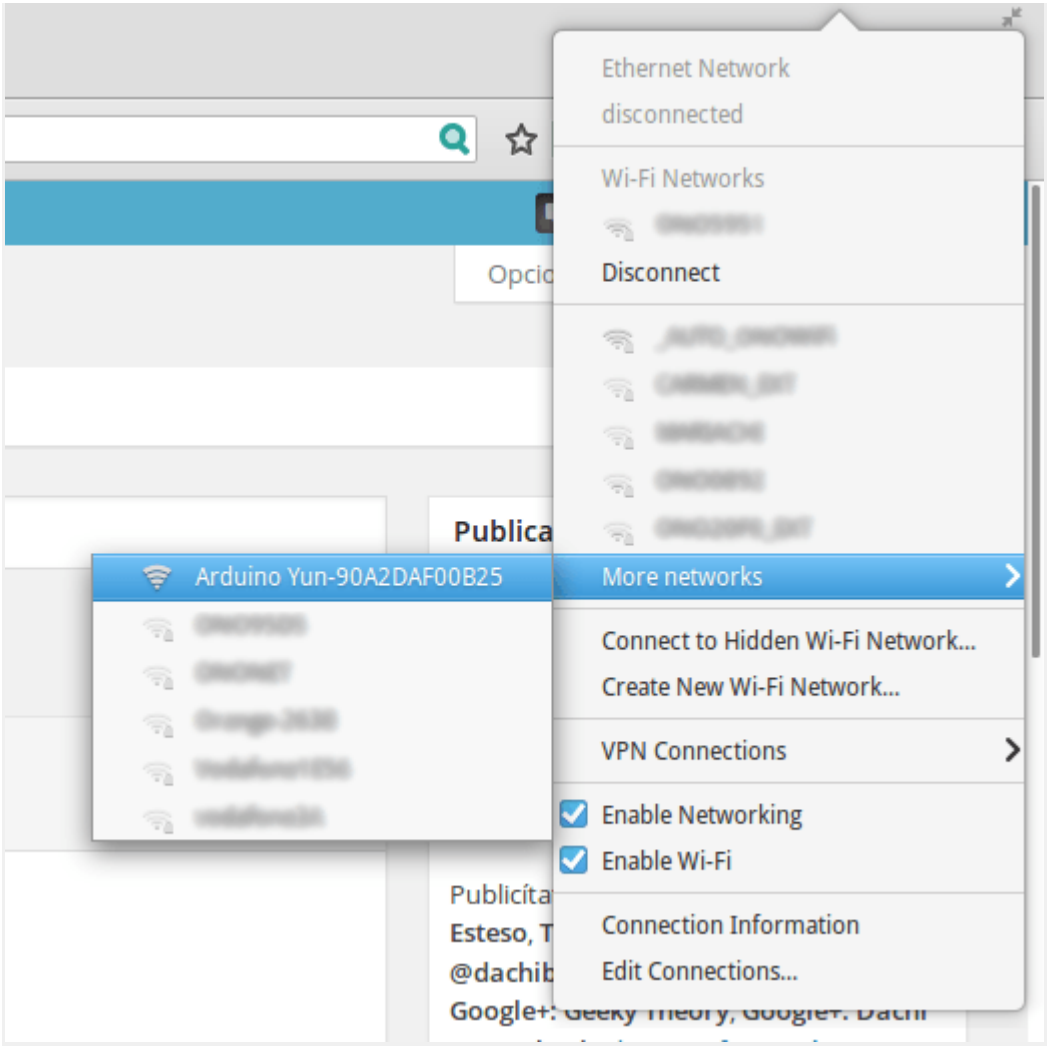


Figura 3.8 Arduino Yun

PASO 2

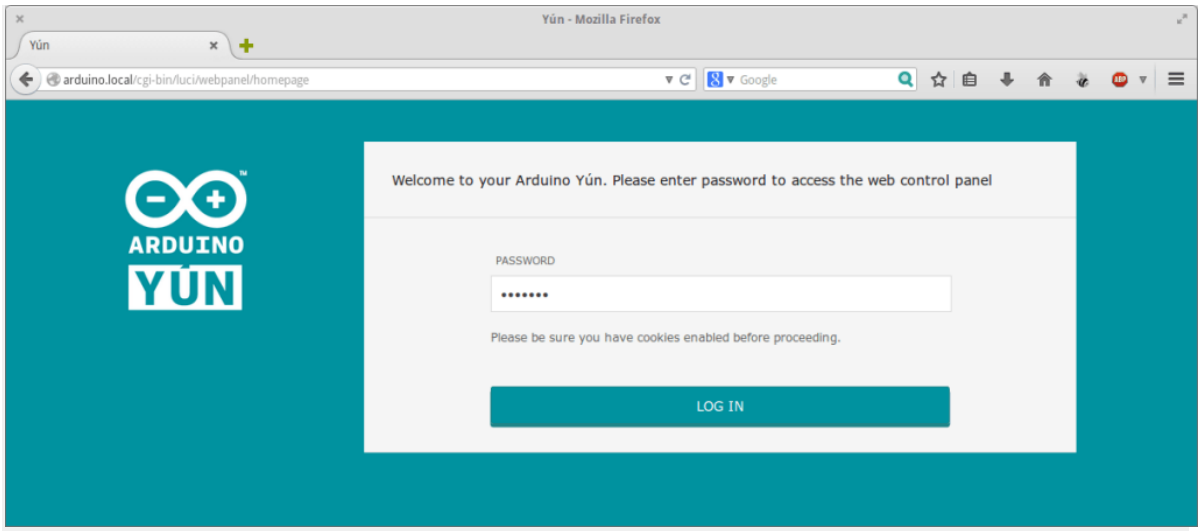
Se encenderá y se verá una nueva red disponible que tiene el prefijo "**Arduino Yun-**". Conectarse a ella:



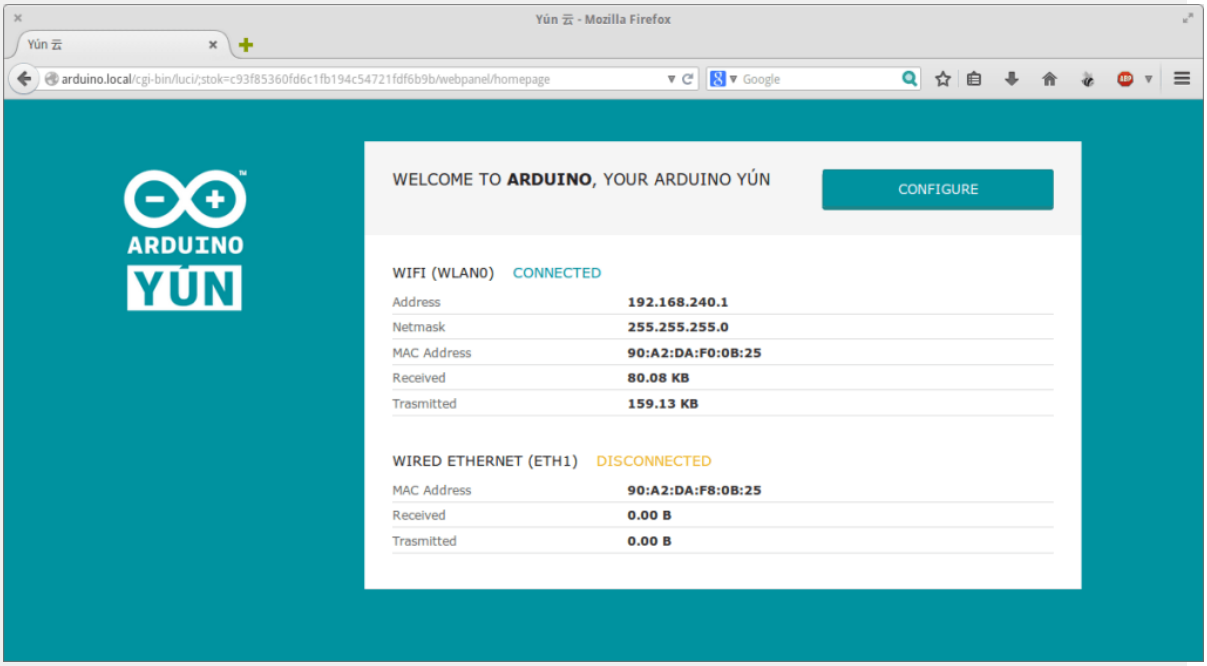
PASO 3

Una vez nos hemos conectado a la red del Arduino Yun, entramos a la Web arduino.local y ya podremos realizar la configuración.

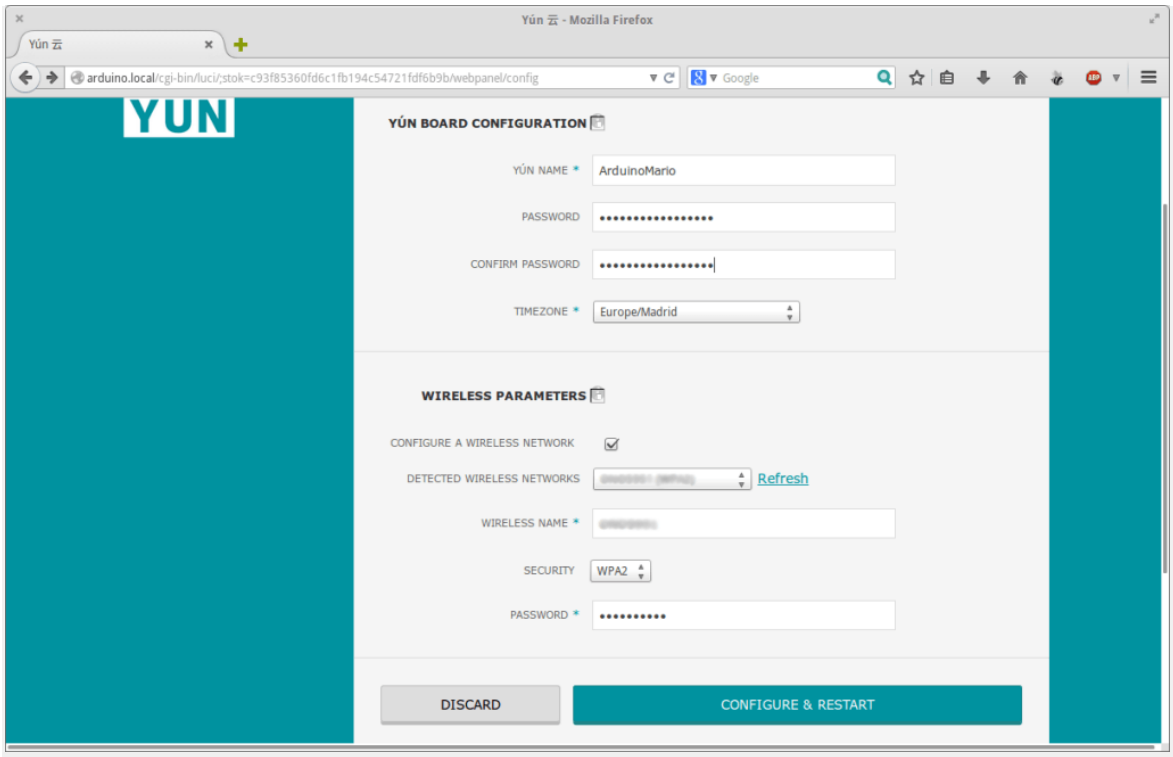
Como contraseña tenemos que poner: **Arduino**



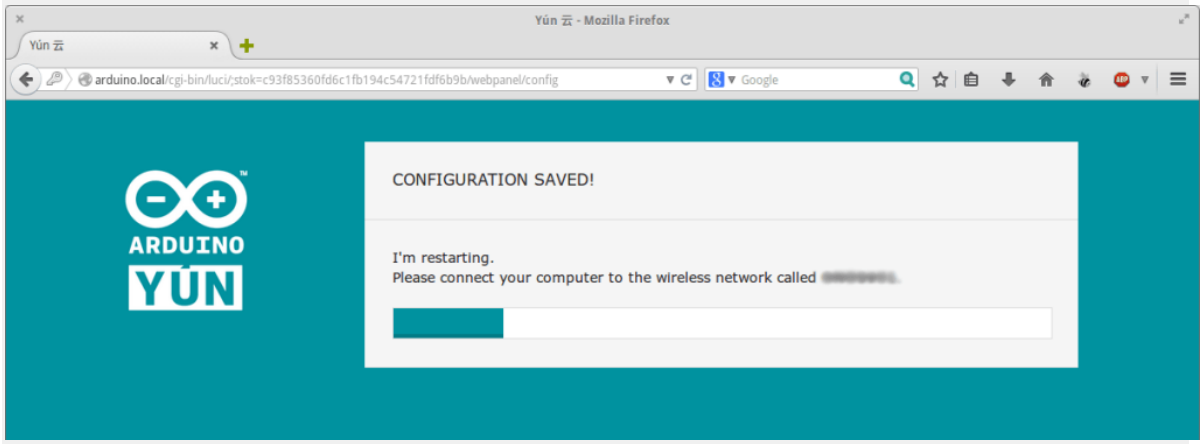
Una vez hemos entrado con la contraseña, nos aparecerá una ventana en la que **podremos ver la información de nuestra placa**:



Vamos a configurarla haciendo click en el botón **CONFIGURE** y le damos al Arduino el nombre que queramos y, también podemos cambiar la contraseña. Configuraremos la red WiFi para que se conecte a ella y poder trabajar con él:



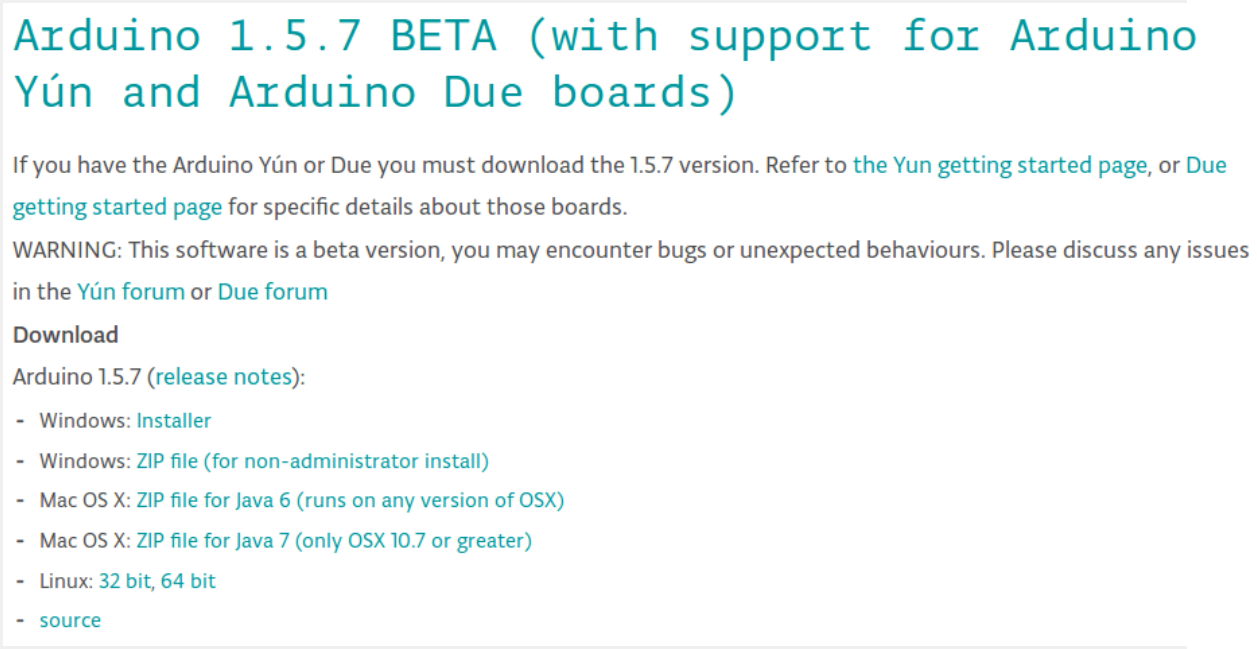
Finalmente, click en **Configure & Restart** para acabar la configuración del Arduino y **reiniciarlo**.



¡Atención! Para volver a entrar a la configuración de Arduino, introducid el nombre que le habéis dado concatenado con ".local". En mi caso sería **arduinomario.local**. Lo digo porque llevo 20 minutos intentando entrar a **arduino.local** sin mucho éxito.

PASO 4: INSTALACIÓN DEL IDE DE ARDUINO

Necesitamos instalar como mínimo la versión 1.5 para poder utilizar el Arduino Yun. Para ello, vamos a este enlace y lo descargamos para la plataforma que nos corresponda. En mi caso es Linux 64bit y a día de hoy es la versión 1.5.7 BETA.



Arduino 1.5.7 BETA (with support for Arduino Yún and Arduino Due boards)

If you have the Arduino Yún or Due you must download the 1.5.7 version. Refer to the Yun getting started page, or Due getting started page for specific details about those boards.

WARNING: This software is a beta version, you may encounter bugs or unexpected behaviours. Please discuss any issues in the Yún forum or Due forum

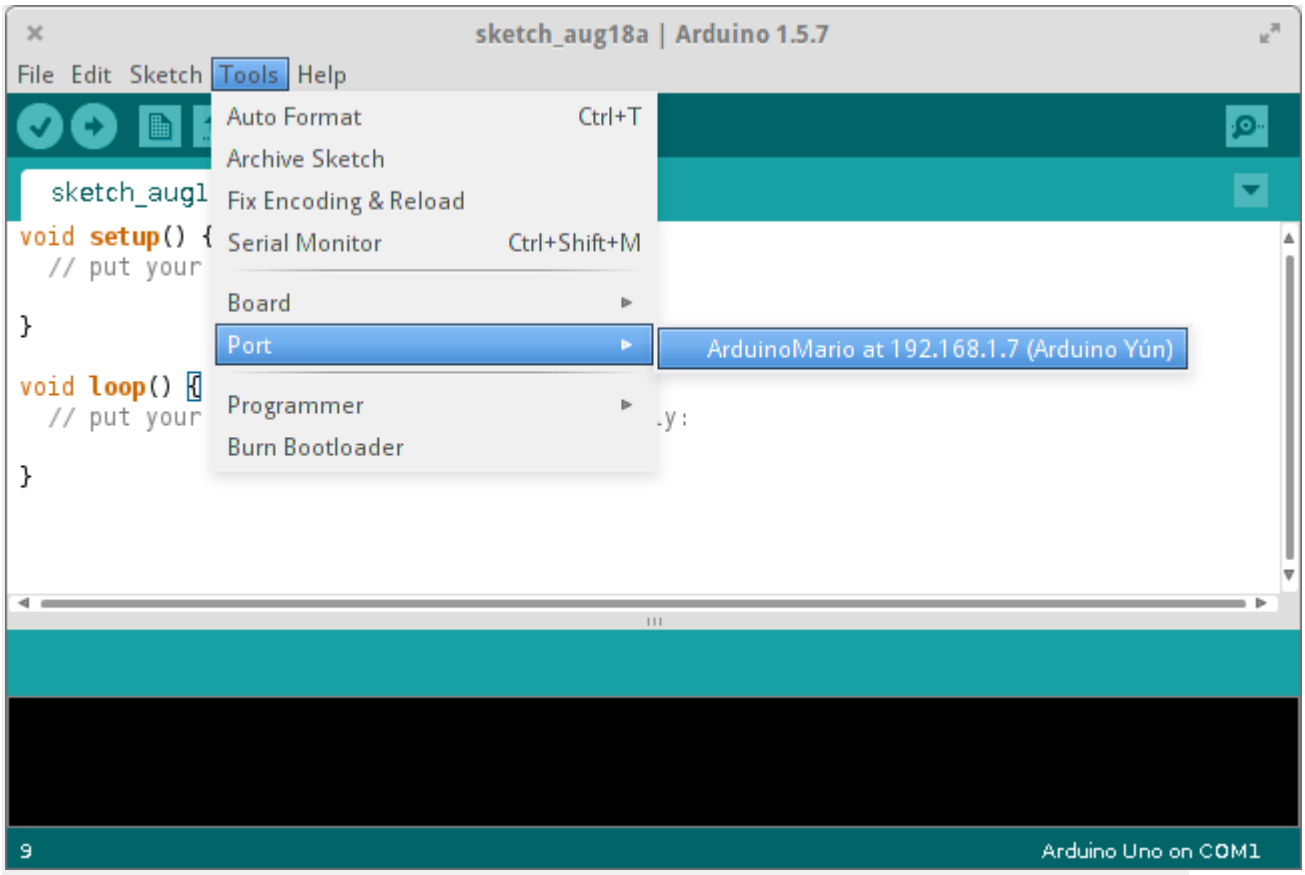
Download

Arduino 1.5.7 (release notes):

- Windows: Installer
- Windows: ZIP file (for non-administrator install)
- Mac OS X: ZIP file for Java 6 (runs on any version of OSX)
- Mac OS X: ZIP file for Java 7 (only OSX 10.7 or greater)
- Linux: 32 bit, 64 bit
- source

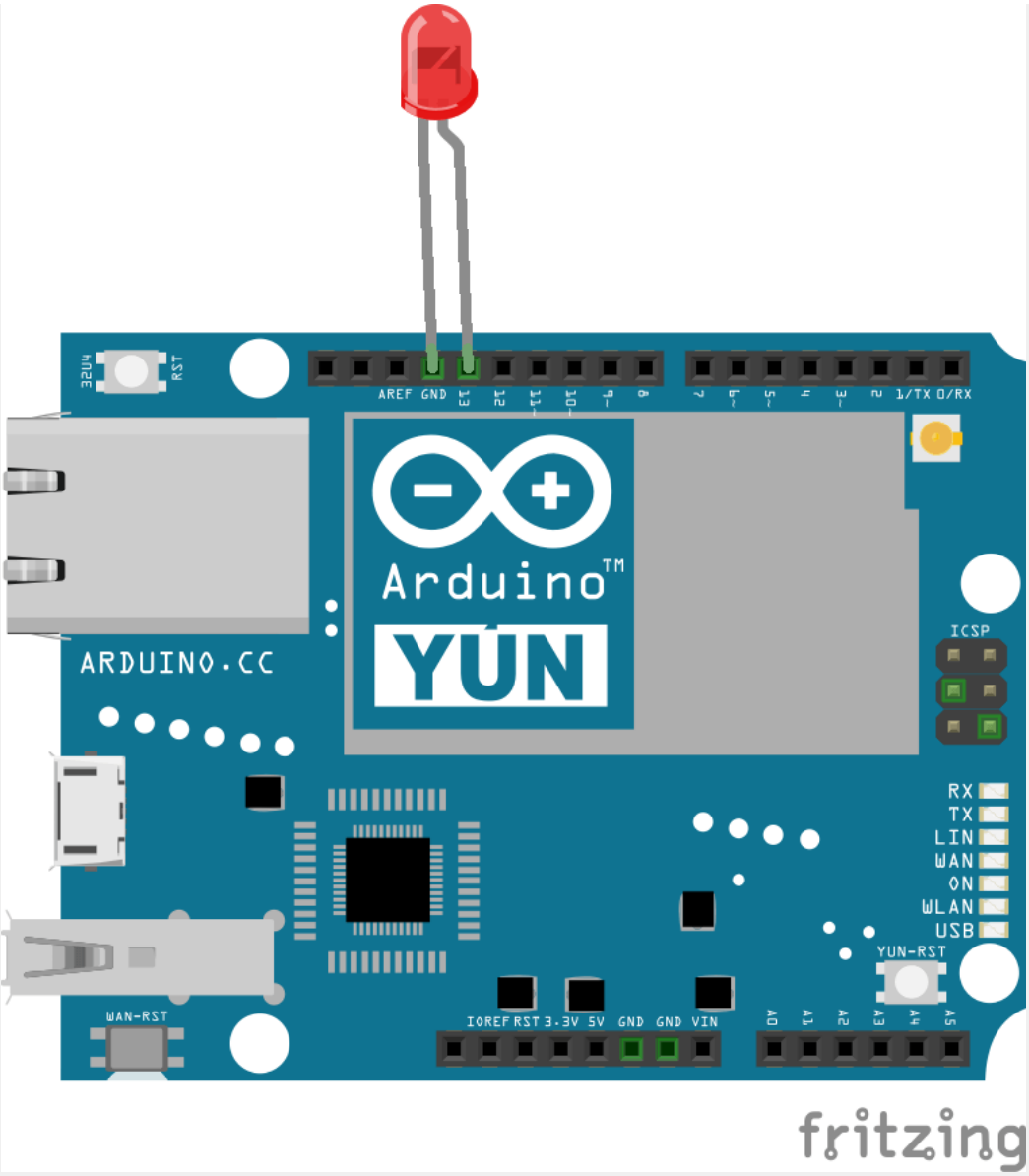
PASO 5: PRIMER PROGRAMA

Abrir el IDE que se acaba de instalar. dirigirse a **Herramientas->Puerto** y seleccionar el Arduino conectado:



En el modelo de placa, seleccionar **Arduino Yun**. Si no, no funcionará.

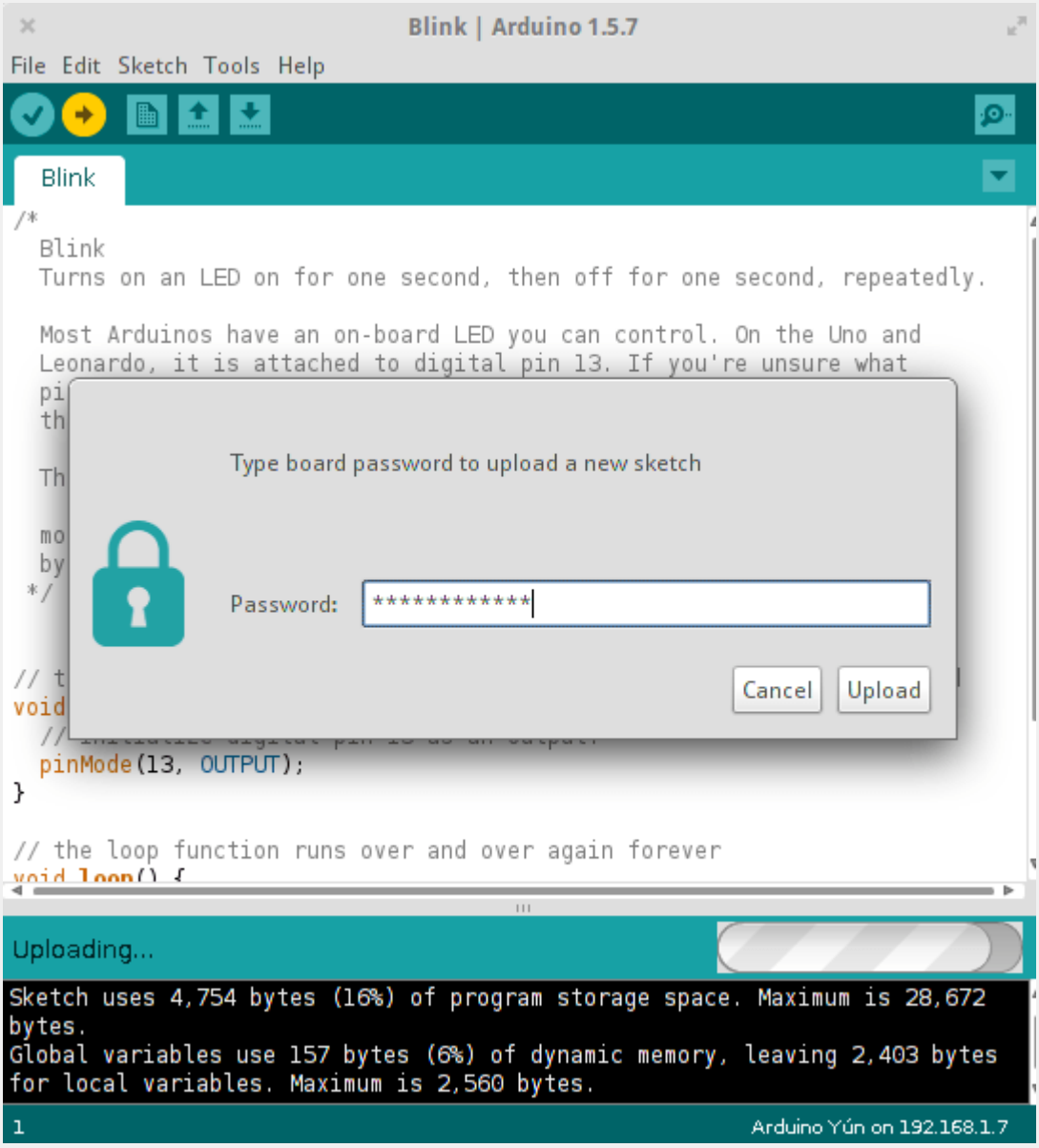
Realizar este montaje, que va a consistir en **un LED conectado al pin número 13 para que parpadee**. El típico ejemplo de siempre.



El código que se utiliza es el siguiente:

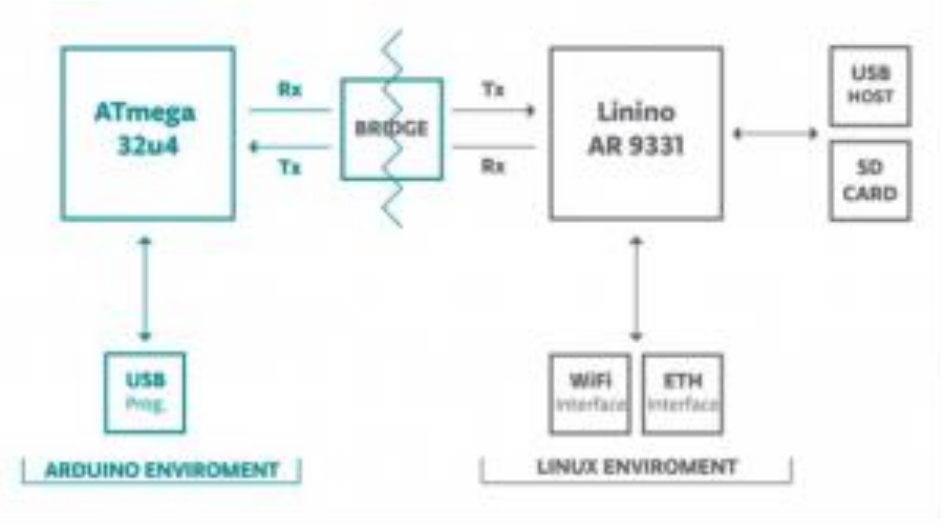
```
    void setup() {
1     pinMode(13, OUTPUT);
2   }
3   void loop() {
4     digitalWrite(13, HIGH);
5     delay(1000);
6     digitalWrite(13, LOW);
7     delay(1000);
8   }
9
```

Lo cargamos y **nos pedirá la contraseña del Arduino**:



Una vez se haya subido el *sketch*, el LED estará parpadeando.

Básicamente, Arduino Yun es una placa que comparte chip y pineado con un Arduino Leonardo con chip ATmega32u4, y una controladora basada en el chip Atheros AR9331, una solución wifi completa corriendo un Linux “Linino” basado en Open-WRT.
En este esquema puede verse mejor:



Esquema 1. Open- WRT

Por buscar una comparación, es como si se tuviera un ya famoso WRT54G, el tan mítico router azul de Linksys, y un Arduino Yun, conectados por un serial cruzado entre ellos. Y eso es exactamente lo que nos ofrece Arduino Yun. Open-WRT tiene una enorme comunidad detrás, paquetes disponibles para casi todo lo que puedas imaginar, y muchos años de experiencia en su desarrollo.

Por eso nos encontramos con que el Yun nos ofrece a mayores de un simple Arduino con conectividad wifi, un servidor web, comunicación bidireccional entre la parte Arduino y el Linux embebido, una api REST para manejar los pines del Arduino, soporte Python 2.7 etc…

Hasta ahí, la cosa ya suena bien, pero es que con unos sencillos comandos podemos aumentar sus funciones de una forma increíble, ya que cuenta con un sistema de paquetes muy fácil de utilizar, el ya famoso opkg.

Podemos darle soporte PHP4 y 5, mysql, hacer streaming de una webcam, soporte para medios extraíbles USB, conectividad 3G USB, instalar una tarjeta de sonido USB… Las posibilidades son infinitas, y abre una nueva era en cuanto a comunicaciones en placas Arduino.

Aunque de un primer vistazo pueda parecer un precio elevado para una placa Arduino, solo hay que hacer una comparación sencilla, pero lo primero es echarle un vistazo y conocerlo un poco mejor:

Figura 3.9 vista frontal del Arduino Yun

Figura 3.10 Vista trasera del Arduino Yun

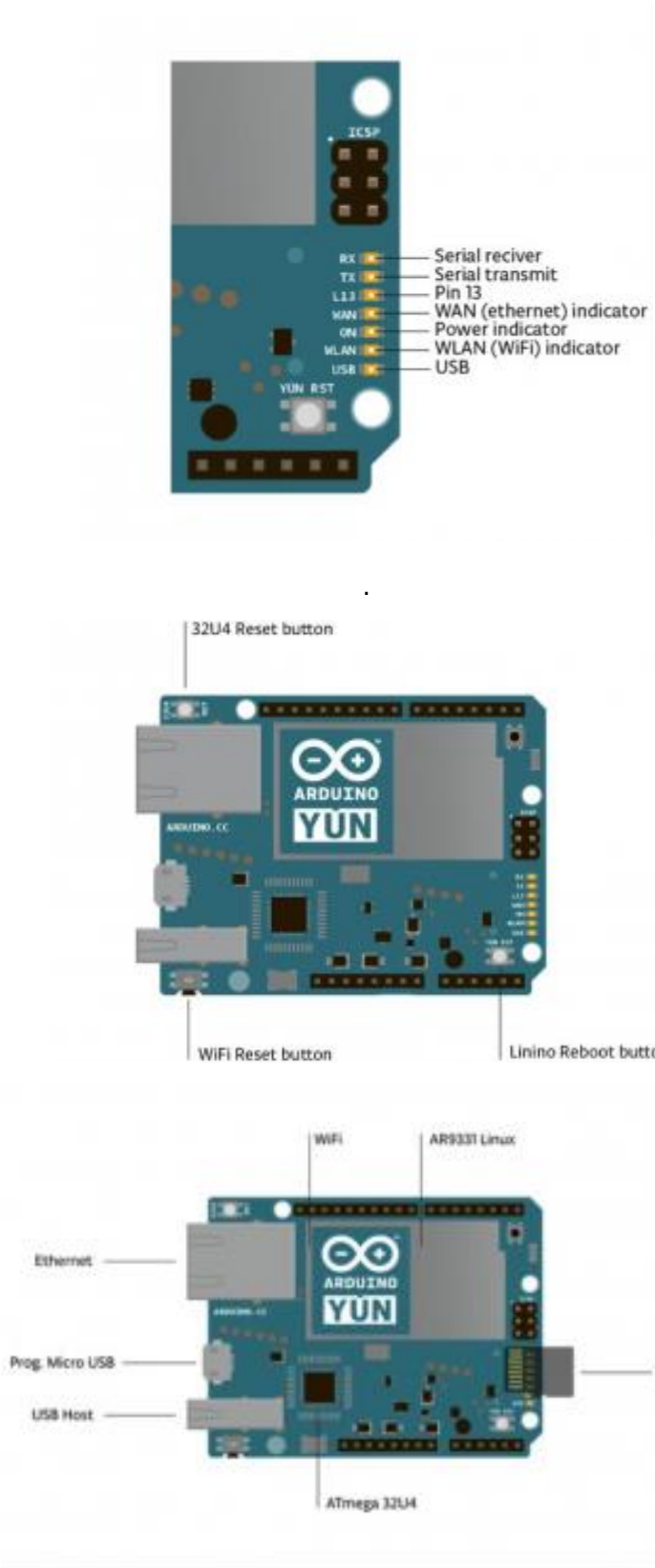


Figura 3.11 Botones, conectores y leds de señalización

El formato seleccionado es el del Arduino UNO / Leonardo, por lo que es compatible con casi todas las shields disponibles en el mercado. En el caso del proyecto se selecciona el Arduino Mega.

Al conectar el Arduino Yun a la alimentación, se creará automáticamente una red inalámbrica Ad-Hoc con un nombre aleatorio del tipo ArduinoYun-XXXXXXXXX sin cifrado, a la que nos conectaremos desde cualquier dispositivo wifi.

Por defecto, viene configurado en la ip 192.168.240.1, y nos dará una ip por dhcp en ese mismo rango.

Abrimos un navegador y apuntamos a esa ip o bien a la URL arduino.local, y nos encontraremos con la página de login de Luci, la interface web de configuración de nuestro Yun, con contraseña por defecto “Arduino”.

En la primera ventana se encuentra con un resumen de las interfaces activas, y las ip de las mismas, interesante si utilizáis dhcp.

Si pulsa el botón de configure, accederá al panel simple de gestión de redes inalámbricas, solo necesita escoger la red en el desplegable y poner la pass, ya detecta la encriptación de la misma al seleccionarla. También puede asignarle un nombre para acceder por el navegador y hacerle peticiones rest, con el formato xxxx.local.

El Arduino YUN se reiniciará, y se conectará automáticamente a la red inalámbrica, momento en el cual aparecerá en nuestro ide, en el apartado de selección de placas un nuevo elemento, y ya podremos subir sketch de forma inalámbrica al YUN, por lo que no tenemos que desconectarlo del proyecto para subir cambios, todo un logro.

3.4 Configuración del proceso de datos a la red.

Para poder configurarlo nos guiamos del siguiente ejemplo:

Configuración de sus cuentas Temboo y Dropbox

Antes de que podamos construir aplicaciones emocionantes con nuestro hardware, necesitamos configurar algunas cuentas en los servicios web que vamos a utilizar. La primera es Temboo, donde se necesita tener una cuenta. Temboo, básicamente, hacer que la interfaz entre el Arduino Yun y Dropbox. Sólo tienes que ir a: https://www.temboo.com/ Se le pedirá que cree una cuenta, y entonces su primera aplicación. Anote el nombre de su cuenta, el nombre de la aplicación y su clave de API de aplicación, necesitará más tarde. También hay una cosa más que necesita de la página web Temboo: la Temboo Python SDK, que utilizaremos más adelante para subir fotos en Dropbox. Puedes conseguirlo en:https://www.temboo.com/python Una vez descargado, simplemente extraer la carpeta en la tarjeta microSD.Entonces, usted necesita una cuenta de Dropbox. Ir a la página web de Dropbox para hacerlo: https://www.dropbox.com/home

3.4.1 Configuración del Arduino Yun

Antes de que podamos escribir el software para nuestro proyecto, tenemos que instalar algún software más en el Arduino Yun. También vamos a probar la cámara USB para ver si los controladores están trabajando correctamente.

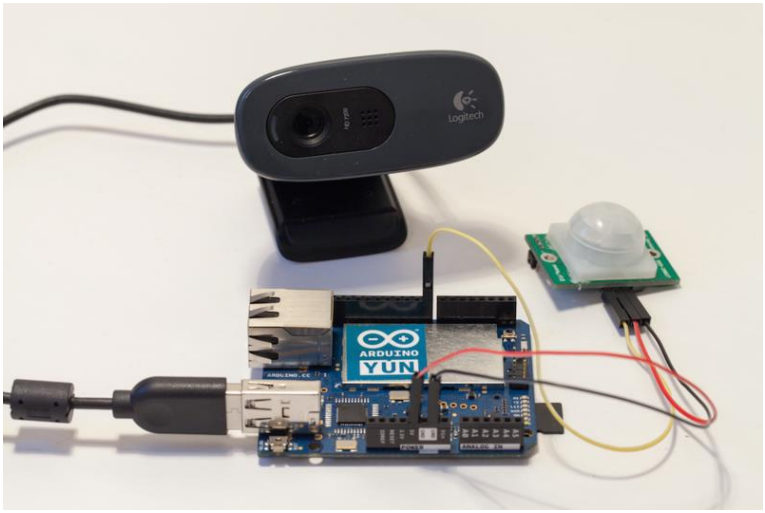


Figura 3.12 cámara y sensor de movimiento conectados al Arduino Yun

[11]En el resto del proyecto, voy a asumir que su Yun ya está configurado y conectado a una red WiFi o LAN local. En primer lugar, los controladores UVC. Para instalarlos, es necesario conectarse a su Yun a través de SSH. Basta con abrir una terminal y escriba:

Cuando usted necesita para insertar el nombre correcto de su Arduino Yun, que estableció la primera vez que utilizó el Yun. También se le pedirá que introduzca su contraseña. Si la conexión se realiza correctamente, debería ver un poco de arte ASCII:

Ahora podemos instalar los paquetes necesarios. Comience con una actualización del gestor de paquetes:

actualización opkg

A continuación, instale los controladores UVC:

```
1. opkg instalar kmod - vídeo - uvc
```

Y el paquete python-openssl:

```
1. opkg instalar python - openssl
```

También necesitamos la utilidad fswebcam que vamos a utilizar para tomar fotos de la terminal:

```
1. opkg instalar fswebcam
```

Para la última parte del proyecto, también vamos a necesitar la biblioteca de streaming mjpg. Usted puede obtener con:

```
1. opkg instalar mjpg - streamer
```

Ahora estamos listos para probar la cámara web. Asegúrese de que la tarjeta SD está montado en el Yun, y vaya a la carpeta de la tarjeta SD con:

```
1. cd / mnt / sda1
```

Para probar la cámara y tomar una foto, es realmente fácil. Simplemente escriba:

```
1. prueba fswebcam. png
```

Usted debe ver un poco de información que se muestra, junto con algunos errores, pero no te preocupes por ellos. Lo importante es ver estas líneas:

```
1.  --- Apertura / dev / video0 ...
2.  Tratando fuente módulo v4l2 ...
3.  / Dev / video0 abrió.
```

Para comprobar que la imagen fue tomada correctamente, retire la tarjeta SD de la Yun y leerlo mediante el ordenador. Usted debe ver la imagen que aparece en la raíz de la tarjeta SD:

Figura 3.11 vista de la información guardada en la SD

Basta abrirlo para asegurarse de que se toma correctamente y que no está dañado. Si la imagen se ve bien, puede pasar a la siguiente sección y empezar a construir aplicaciones de frío con el proyecto!

Sube imágenes a Dropbox

Lo que queremos lograr en esta primera aplicación es para tomar una imagen cada vez que algún movimiento es detectado por el sensor de movimiento PIR. Y cuando esto sucede, almacenar esta imagen localmente en la tarjeta SD, y subirlo a Dropbox. Para ello, el código se compone de dos partes. El primero es un script en Python que se conectará a Dropbox, tomar una foto en la tarjeta SD, y luego subir la imagen a Dropbox. La razón para utilizar Python para esta parte es que es mucho más fácil de subir archivos a Dropbox usando Python que directamente desde el boceto Arduino. La segunda parte del código será el Arduino dibujar en sí, que, básicamente, llame a la secuencia de comandos de Python para tomar imágenes a través de la biblioteca de Puente del Yun. Del primer código Deje la secuencia de comandos de Python. Se inicia mediante la inclusión de las bibliotecas necesarias desde el SDK Temboo Python:

```
1.  desde temboo. núcleo. sesión de importación  TembooSession
2.  desde temboo. Biblioteca. Dropbox. FilesAndMetadata  importación  UploadFile
```

La secuencia de comandos de Python también tomará el nombre de la imagen que queremos subir como argumento:

```
1.  con abiertas (str (sys. argv [1]),  "rb")  como archivoDeImagen:
2.    Encoded_String = base64. b64encode (archivoDeImagen. leer ())
```

Recuerde estas credenciales Temboo que creó anteriormente? Ahí es donde usted necesita para entrar en ellos:

```
1. sesión =  TembooSession ('yourTembooName',  'yourTembooApp',
   'yourTembooKey')
```

A continuación, puede crear la biblioteca Dropbox correcta para subir archivos, llamado "Choreo" en Temboo:

```
1. uploadFileChoreo =  UploadFile (sesión)
2. uploadFileInputs = uploadFileChoreo. new_input_set ()
```

Ahora es el momento de entrar en todas las informaciones sobre su cuenta de Dropbox, como la clave de aplicación, aplicación Secreto, Acceso Token un Access Token Secret:

```
1. uploadFileInputs. set_AppSecret ("appSecret")
2. uploadFileInputs. set_AccessToken ("accessToken")
3. uploadFileInputs. set_FileName (str (sys. argv [1]))
4. uploadFileInputs. set_AccessTokenSecret ("accessTokenSecret")
5. uploadFileInputs. set_AppKey ("appKey")
6. uploadFileInputs. set_FileContents (Encoded_String)
7. uploadFileInputs. set_Root ("caja de arena")
```

Y, por último, cargar el archivo en Dropbox:

```
1. uploadFileResults = uploadFileChoreo. execute_with_results (uploadFileInputs)
```

Ahora puede guardar el código en un archivo llamado upload_picture.py. Tenga en cuenta que todos los archivos están disponibles en el repositorio GitHub del proyecto:https://github.com/openhomeautomation/arduino-yun-camera ahora Vamos a trabajar en el sketch de Arduino. El boceto se inicia mediante la inclusión de las bibliotecas necesarias:

```
1. #include  <Bridge.h>
2. #include  <Process.h>
```

También tenemos que declarar un proceso, que vamos a utilizar para llamar a funciones en la máquina Linux del Yun (por ejemplo, la utilidad fswebcam hemos utilizado antes):

```
1. Proceso de imagen;
```

También vamos a crear un nombre de archivo para cada imagen del proyecto se llevará a, que se almacena en una cadena:

```
1. Cadena de nombre de archivo;
```

También declaramos que el pasador en la que está conectado el sensor de movimiento PIR:

```
1. int pir_pin =  8;
```

Y el camino de la tarjeta SD en el Yun:

```
1. Cadena ruta =  "/ mnt / sda1 /";
```

Porque tenemos que llamar a funciones en la máquina Linux del Yun, tenemos que empezar el puente:

```
1. Puente. Iniciar ();
```

Luego, en la parte de bucle () del boceto, comprobamos si algún movimiento fue detectado por el sensor PIR:

```
1. si  (digitalRead (pir_pin)  ==  verdad)  {
```

Si este es el caso, construimos un nombre de archivo único para la imagen, con la fecha en que fue tomada la imagen:

```
1. nombre =  "";
2. foto. runShellCommand ("date +% s");
3. mientras que (la imagen. correr ());
4.
5. mientras que  (la imagen. disponible ()> 0)  {
6.   Char c = foto. leer ();
7.   nombre de archivo + = c;
8.   }
9. nombre de archivo. recortar ();
10. nombre de archivo + =  ".png";
```

Entonces hacemos la primera llamada a la máquina Linux del Yun, primero para hacer una foto con la utilidad fswebcam. Tenga en cuenta que aquí, ofrecemos un argumento extra con el comando -r, que establece la resolución. Usé la resolución máxima de mi cámara, que es 720p:

```
1. foto. runShellCommand ("fswebcam"  + ruta + nombre +  "-r 1280x720");
2. mientras que (la imagen. correr ());
```

A continuación, hacemos una segunda llamada a la máquina Linux, esta vez de llamar a la secuencia de comandos de Python con el nombre de la imagen como un argumento, que cargar la imagen a Dropbox:

```
1. foto. runShellCommand ("python"  + ruta +  "upload_picture.py"  + ruta + nombre de
   fichero);
2. mientras que (la imagen. correr ());
```

Ahora está listo para probar el proyecto. Una vez más, todos los archivos están disponibles en el repositorio GitHub del proyecto: https://github.com/openhomeautomation/arduino-yun-camera En primer lugar, poner el archivo de Python en la raíz de la tarjeta SD, y poner la tarjeta SD de vuelta en la placa Arduino Yun. A continuación, subir el boceto Arduino a la Yun. Ahora, trata de activar el sensor de movimiento, por ejemplo mediante la renuncia a su mano delante de ella. Usted debe ver que la cámara web está siendo activado poco después (por ejemplo, mi webcam tiene un LED que cambia a verde cuando está activo). Para comprobar que el proyecto funciona correctamente, después de un tiempo se puede comprobar la tarjeta SD, que debiera ver que algunas imágenes se grabaron:

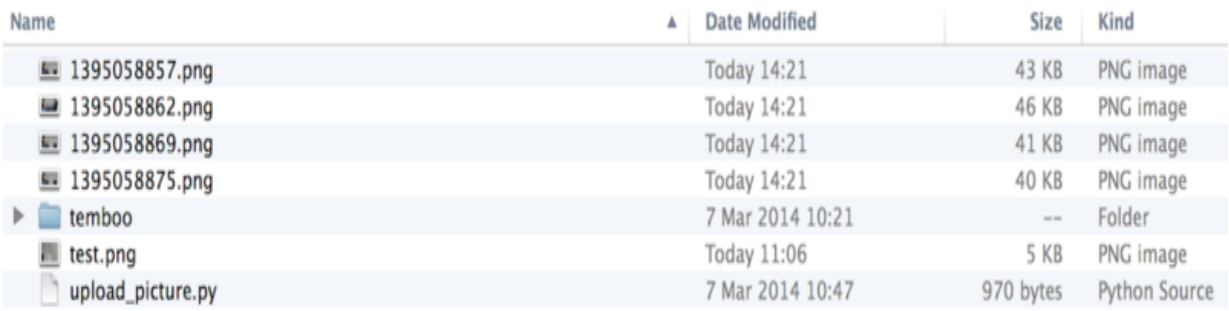


figura 3.13 Imágenes guardadas en la SD durante la prueba.

También se puede consultar en su carpeta de Dropbox, donde se deberían haber subido las mismas imágenes. Deben estar ubicados en su Dropbox apps de carpeta:

Figura 3.14 carpeta del Dropbox

Corriente del vídeo a YouTube

En esta última parte del proyecto, vamos a construir una aplicación completamente diferente para nuestro hardware: hacer vivir la secuencia de vídeo de la cámara en YouTube. Estamos primero vamos a hacer que el flujo de vídeo de la cámara a nivel local, a continuación, transmitir esta corriente al ordenador mediante un software llamado wirecast, que finalmente transmitir el vídeo a un evento en vivo de Youtube.Streaming de vídeo a nivel local es realmente muy fácil, gracias al software que instalado antes. Sólo tiene que acceder a la Yun de nuevo a través de SSH y escriba:

```
1. mjpg_streamer - i "input_uvc.so -d / dev / video0 -r 640x480 -f 25" - o "output_http.so -p 8080 w / www / webcam" y
```

Algunas explicaciones acerca de este código: el argumento -r establece la resolución (yo usé una resolución más baja aquí porque el streaming de vídeo de alta definición se estaba quedando) y el argumento -p establece el puerto en el que la corriente estará disponible. Para ver realmente la corriente, sólo tiene que ir ha (reemplazando el nombre de tu placa Arduino Yun):

```
1. http: //myarduinoyun.local:8080/stream.html
```

Usted debe obtener una página que es la interfaz a la corriente en su Arduino Yun. En esta página, usted debe ver el video de la cámara se transmiten en vivo:

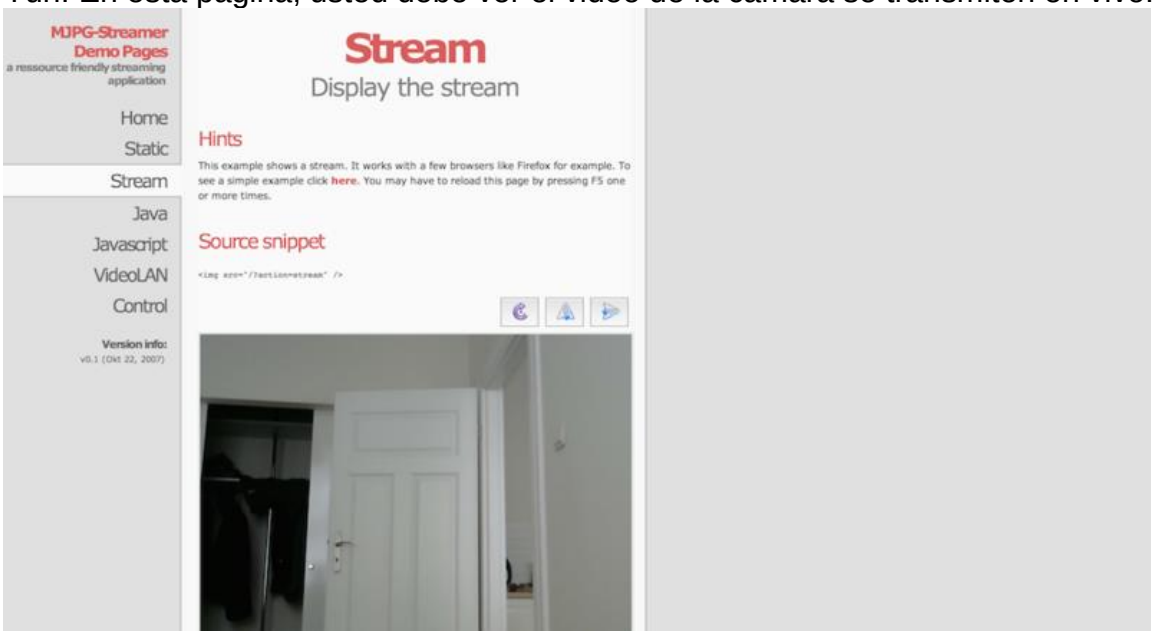


Figura 3.15 vista de la página por la que se ve la imagen de la cámara conectada al Arduino

Ahora, vamos a enviar esta corriente en YouTube. Empieza por ir a la interfaz de YouTube, y la creación de un evento en vivo:

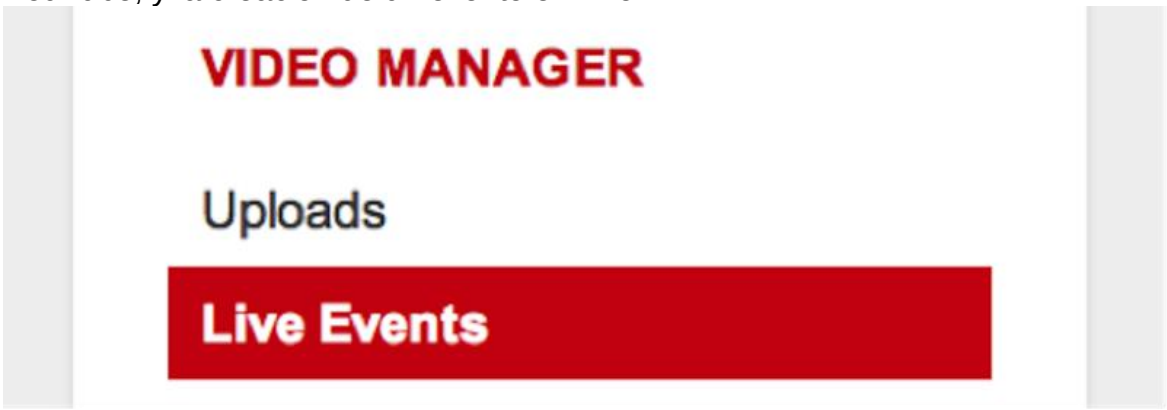


Figura 3.16 creación de un evento en vivo de YouTube.

YouTube le pedirá alguna información acerca de su corriente, y recomendar algún tipo de software para que el ordenador pueda transmitir a YouTube. Solía wirecast, pero debería funcionar con otro software de streaming también. En wirecast, es necesario agregar una nueva "corriente Web" con los siguientes parámetros: el protocolo HTTP, formato Motion JPEG, y "nameofyourYun.local: 8080 / acción = corriente "como el URI. Esta imagen muestra los parámetros que he utilizado:

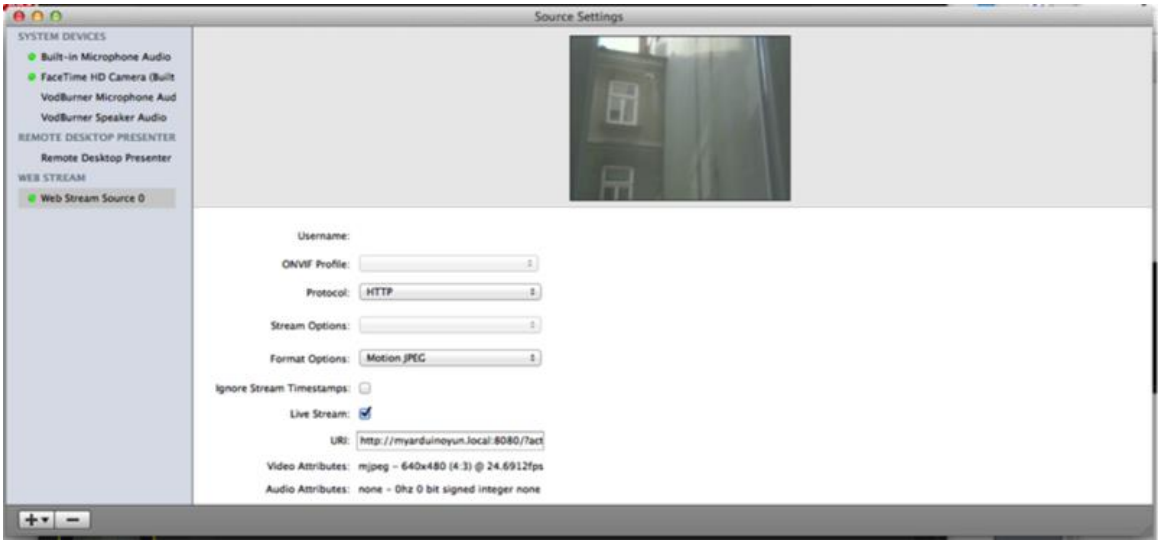


Figura 3.17 configuración de la pagina

La versión gratuita de wirecast insertará algunas marcas de agua en el arroyo, como la corriente de Web es una función de pago. Pero eso no es realmente un

problema para vigilar su casa, ya que no aparecen todo el tiempo. Después de un momento, la corriente de su Yun debe aparecer en la interfaz wirecast:

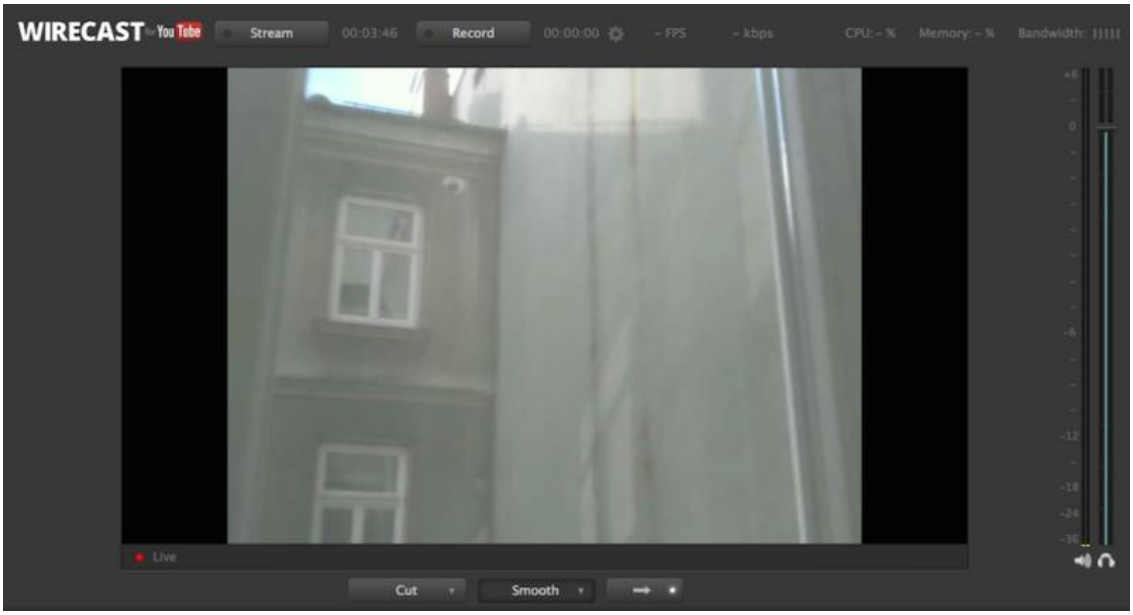


Figura 3.18 vista de la cámara desde la web

Todavía en wirecast, ahora se puede pulsar sobre "Stream". El software le pedirá las credenciales de YouTube, y debería detectar automáticamente su evento en vivo. Una vez hecho esto, se puede volver al evento en vivo en YouTube. En la sala de control de eventos en vivo, usted debe ver que YouTube está recibiendo algunos datos:

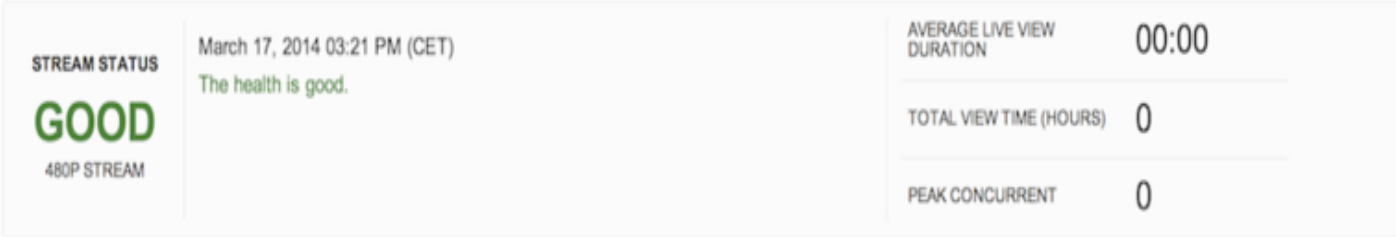


Figura 3.19 muestra de que el servidor de YouTube está recibiendo datos

Si usted puede ver el estado "bueno", puede hacer clic en "Vista previa" para YouTube puede preparar su corriente:

Figura 3.20 muestra la vista previa si el estado de envío de datos es bueno

Tomará un tiempo (algunos minutos en mi caso), pero en algún momento usted será capaz de hacer clic en "Iniciar la transmisión". Una vez hecho esto, puede acceder a su torrente igual que lo haría acceder a cualquier vídeo de YouTube:

Figura 3.21 vista de la cámara desde YouTube.

Enhorabuena, ahora puede monitorear su casa desde cualquier lugar del mundo, sólo por tener la URL a este evento en vivo YouTube! Usted tiene varias opciones para crear su evento en vivo, pero me puse la mía "no listado" para que pueda compartir la dirección con la gente que confío, por lo que también puede tener una mirada en mi casa cuando no estoy allí.

Código de ejemplo:

```
desde temboo. núcleo. sesión de importación  TembooSession
desde temboo. Biblioteca. Dropbox. FilesAndMetadata  importación
con abiertas (str (sys. argv [1]),  "rb")  como archivoDeImagen:
  Encoded_String = base64. b64encode (archivoDeImagen. leer ())
sesión =  TembooSession ('yourTembooName',  'yourTembooApp',  'yourTembooKey')
uploadFileChoreo =  UploadFile (sesión)
uploadFileInputs = uploadFileChoreo. new_input_set ()
uploadFileInputs. set_AppSecret ("appSecret")
uploadFileInputs. set_AccessToken ("accessToken")
uploadFileInputs. set_FileName (str (sys. argv [1]))
uploadFileInputs. set_AccessTokenSecret ("accessTokenSecret")
uploadFileInputs. set_AppKey ("appKey")
uploadFileInputs. set_FileContents (Encoded_String)
uploadFileInputs. set_Root ("caja de arena")
uploadFileResults = uploadFileChoreo. execute_with_results (uploadFileInputs)
#include  <Bridge.h>
#include  <Process.h>
Proceso de imagen;
int pir_pin =  8;
Cadena ruta =  "/ mnt / sda1 /";
Puente. Iniciar ();
si  (digitalRead (pir_pin)  ==  verdad)  {
nombre =  "";
foto. runShellCommand ("date +% s");
mientras que (la imagen. correr ());

mientras que  (la imagen. disponible ()> 0)  {
  Char c = foto. leer ();
  nombre de archivo + = c;
  }
nombre de archivo. recortar ();{
nombre de archivo + =  ".png";
foto. runShellCommand ("fswebcam"  + ruta + nombre +  "-r 1280x720");
mientras que (la imagen. correr ());
foto. runShellCommand ("python"  + ruta +  "upload_picture.py"  + ruta + nombre de fichero);
mientras que (la imagen. correr ());
}
```

3.5 RESULTADOS

Con la finalidad de determinar el funcionamiento adecuado del sistema de monitoreo y diagnóstico, es necesario realizar pruebas del funcionamiento de los sensores.
Se realizó el algoritmo de adquisición y envió de datos en la plataforma de arduino.1.0.5 que cumpliera con los requerimientos de procesamiento de valores analógicos, digitales y entregue mediciones confiables, por tal motivo de realizo su caracterización en un día despejado.

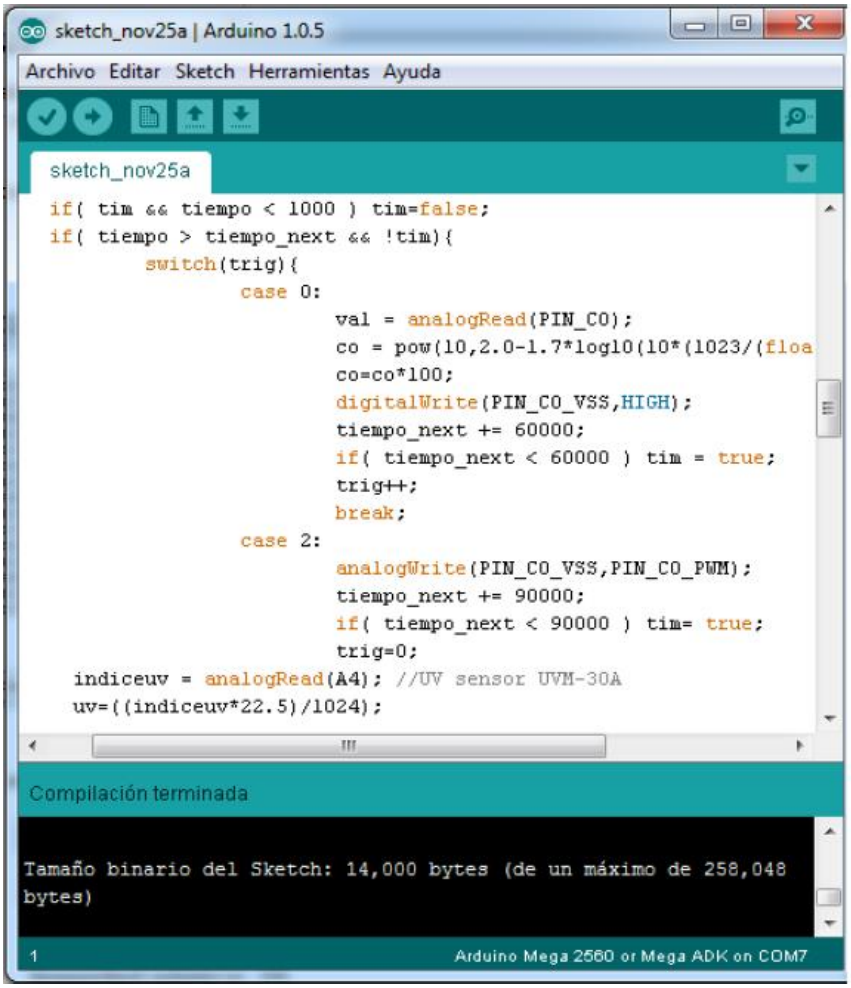


Figura 3.22 Código de prueba de sensores

A continuación se muestra los valores de las pruebas realizadas del dispositivo obteniendo las siguientes mediciones realizadas en la Figura 4.1

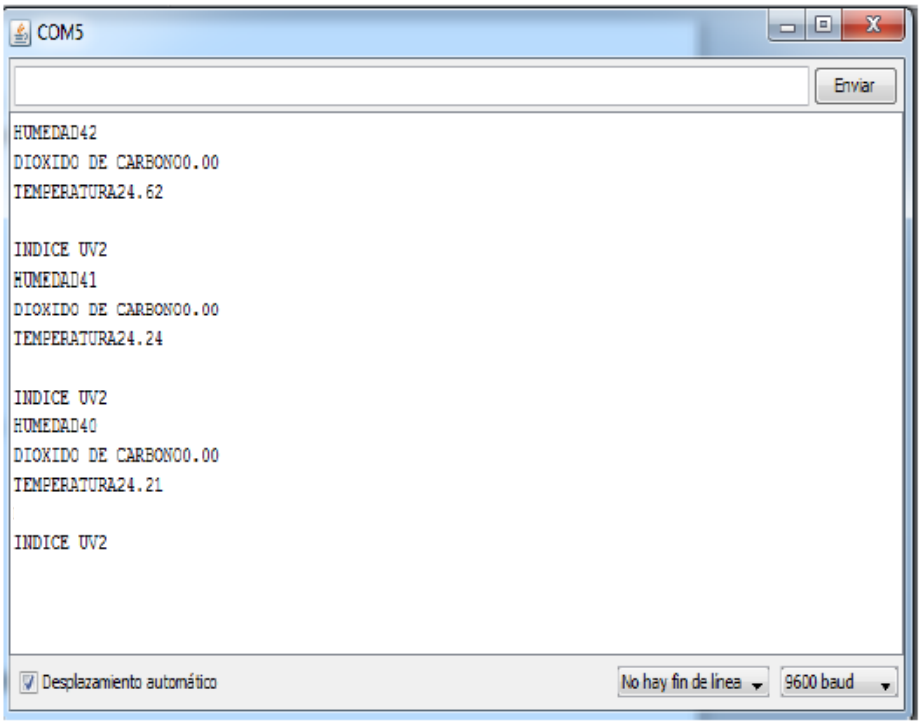


Figura 3.23 Monitor serial de datos enviados en la plataforma Arduino

El envío de datos se hizo por medio de un modem TP-Link genérico de forma alámbrica como se observa en la figura 4.2

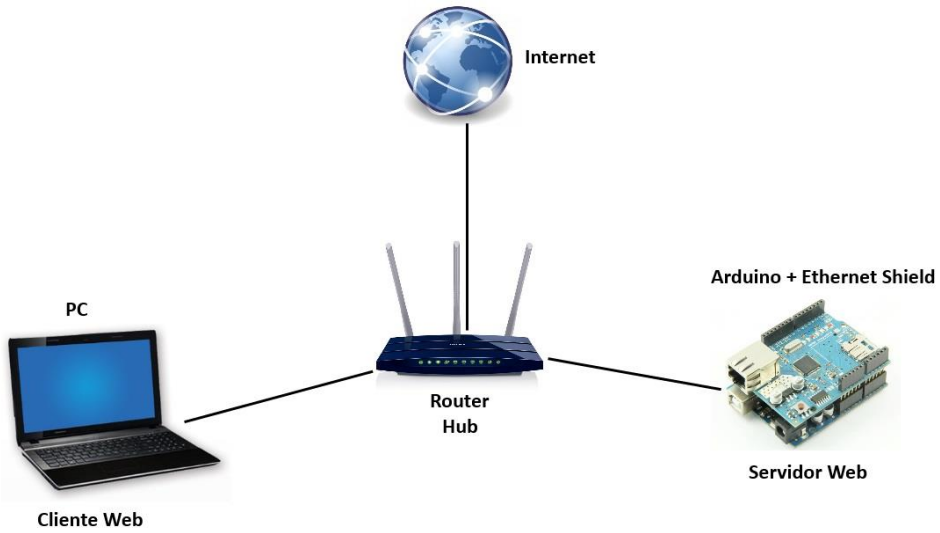


Figura 3.24 visualización del envío de datos

Se realizó un algoritmo en el cual se hace la comunicación de Arduino con la tarjeta Ethernet Shield obteniendo como resultado los valores que se presentan en la figura 3.25.

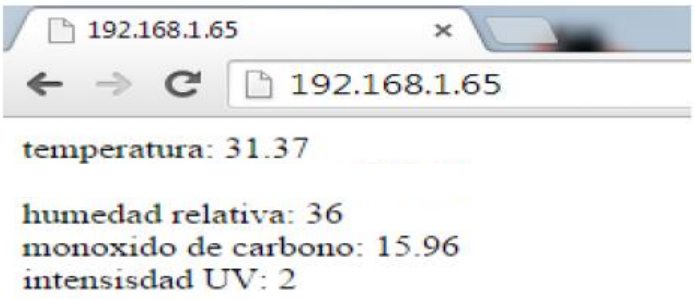


Figura 3.25 Monitor desde la página web

Almacenamiento de datos en Excel

Teniendo la necesidad de poder tener disponible los valores para una futura consulta se propone realizar este almacenamiento a través de Excel, incorporando para cada medición la hora.

Esta tarea se re realiza guardando en primer lugar los valores de los sensores en un archivo de texto, para poder realizar esta captura se utiliza el software libre NetBeans IDE 8.0.2 que es capaz de capturar los datos que son mandados a través del puerto serie.

Posteriormente se importan los datos del archivo de texto a Excel donde es necesario utilizar un marco para poder insertar la hora de las mediciones.

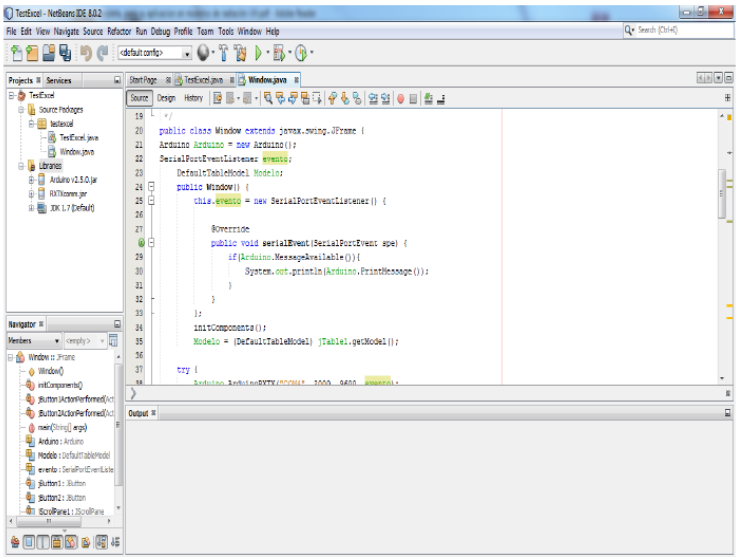


Figura 3.26 software NetBeans

Código en software Netbeans:

```
package testexcel;
import Arduino.Arduino;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import java.io.FileOutputStream;
import java.lang.Thread.State;
import java.util.Calendar;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.table.DefaultTableModel;
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
public class Window extends javax.swing.JFrame {
Arduino Arduino = new Arduino();
int slot=1;
double humedad=0;
double temperatura=0;
double monoxido=0;
double intensidaduv=0;
int lecturas=0;
Calendar Calendario = Calendar.getInstance();
SerialPortEventListener evento = new SerialPortEventListener() {
@Override
public void serialEvent(SerialPortEvent spe) {
if(Arduino.MessageAvailable()){
if(slot==1){
if(lecturas>1){
TableUpdate();
}
slot=2;
lecturas++;
temperatura=Double.parseDouble(Arduino.PrintMessage());
}
else if(slot==2){ slot=3; humedad=Double.parseDouble(Arduino.PrintMessage());
}
```

```
else if(slot==4){ slot=5; monoxido=Double.parseDouble(Arduino.PrintMessage());
}
else if(slot==5){ slot=1;
intensidaduv=Double.parseDouble(Arduino.PrintMessage());
}
}}
};
DefaultTableModel Modelo;
boolean State = false;
public void TableUpdate(){
String Output = "";
String hora = Calendario.get(Calendar.HOUR_OF_DAY) + "";
String minuto = Calendario.get(Calendar.MINUTE) + "";
String segundos = Calendario.get(Calendar.SECOND) + "";
if (Integer.parseInt(hora) < 10) {
hora = "0" + hora;
}
if (Integer.parseInt(minuto) < 10) {
minuto = "0" + minuto;
}
if (Integer.parseInt(segundos) < 10) {
segundos = "0" + segundos;
}
Output = hora + ":" + minuto + ":" + segundos;
Calendario = Calendar.getInstance();
Modelo.addRow(new Object[]{Output+"",temperatura,humedad,
monoxido,intensidaduv});
}
public Window() {
initComponents();
Modelo = (DefaultTableModel) jTable1.getModel();
try {
Arduino.ArduinoRXTX("COM7", 2000, 9600, evento);
} catch (Exception ex) {
Logger.getLogger(Window.class.getName()).log(Level.SEVERE, null, ex);
}
}
@SuppressWarnings("unchecked")
private void initComponents() {
```

```
jScrollPane1 = new javax.swing.JScrollPane();
jTable1 = new javax.swing.JTable();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
jTable1.setModel(new javax.swing.table.DefaultTableModel(
new Object [][] {
},
new String [] {
"Hora", "Temperatura", "Humedad", "CO2", "Indice UV"
}
));
jScrollPane1.setViewportView(jTable1);
jButton1.setText("capturar datos");
jButton1.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
jButton1ActionPerformed(evt);
}
});
jButton2.setText("exportar a excel");
jButton2.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
jButton2ActionPerformed(evt);
}
});
javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addGap(153, 153, 153)
.addComponent(jButton1)
.addGap(126, 126, 126)
.addComponent(jButton2)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
.addGroup(layout.createSequentialGroup()
.addContainerGap()
```

```
.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 654,
Short.MAX_VALUE)
.addContainerGap())
);
layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addContainerGap()
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 253,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASEL
INE)
.addComponent(jButton2)
.addComponent(jButton1))
.addContainerGap(18, Short.MAX_VALUE))
);
pack();
}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
if (State == true) {
State = false;
try {
Arduino.SendData("0");
} catch (Exception ex) {
Logger.getLogger(Window.class.getName()).log(Level.SEVERE, null, ex);
}
} else {
State = true;
try {
Arduino.SendData("1");
} catch (Exception ex) {
Logger.getLogger(Window.class.getName()).log(Level.SEVERE, null, ex);
}
}
}
public void FicheroExcel(String input) {
HSSFWorkbook libro = new HSSFWorkbook();
HSSFSheet hoja = libro.createSheet();
HSSFRow fila = hoja.createRow(0);
HSSFCell celda = fila.createCell(0);
```

```
celda.setCellValue("base de datos"); //tÃtulo
fila = hoja.createRow(1);
celda = fila.createCell(0);
celda.setCellValue("HORA");
celda = fila.createCell(1);
celda.setCellValue("TEMPERATURA");
celda = fila.createCell(2);
celda.setCellValue("HUMEDAD");
celda = fila.createCell(3);
celda.setCellValue("CO2");
celda = fila.createCell(5);
celda.setCellValue("INDICE UV");
for (int i = 0; i <= Modelo.getRowCount() - 1; i++) {
fila = hoja.createRow(i + 2); //se crea la fila
for (int j = 0; j <= 5; j++) {
celda = fila.createCell(j); //se crea la celda
celda.setCellValue(jTable1.getValueAt(i, j).toString()); //se le asigna el valor
}
}
try {
FileOutputStream Fichero = new FileOutputStream(input);
libro.write(Fichero);
Fichero.close();
} catch (Exception e) {
e.printStackTrace();
}
}
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
javax.swing.JFileChooser Ventana = new javax.swing.JFileChooser();
String ruta = "";
try {
if (Ventana.showSaveDialog(null) == Ventana.APPROVE_OPTION) {
ruta = Ventana.getSelectedFile().getAbsolutePath() + ".xls";
FicheroExcel(ruta);
}
} catch (Exception ex) {
ex.printStackTrace();
}
}
public static void main(String args[]) {
try {
for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels())
}
}
} catch (ClassNotFoundException ex) {
```

```
java.util.logging.Logger.getLogger(Window.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
} catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(Window.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(Window.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Window.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
}
java.awt.EventQueue.invokeLater(new Runnable() {
public void run() {
new Window().setVisible(true);
}
});
}
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
// End of variables declaration
}
```

```
if ("Nimbus".equals(info.getName())) {
javax.swing.UIManager.setLookAndFeel(info.getClassName());
break;
}
}
} catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(Window.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
} catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(Window.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(Window.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Window.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
}
java.awt.EventQueue.invokeLater(new Runnable() {
public void run() {
new Window().setVisible(true);
}
});
}
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
// End of variables declaration
}
```

En el código se tiene como finalidad guardar la hora a la que se introduce un valor en cada celda, la hora en la columna A automáticamente el valor del cada sensor en la siguientes columnas (B, C, D, E, F).

En la figura (Figura 3.27) se muestra la interfaz para el almacenamiento de datos en Excel, así también se observan los datos obtenidos por cada sensor.

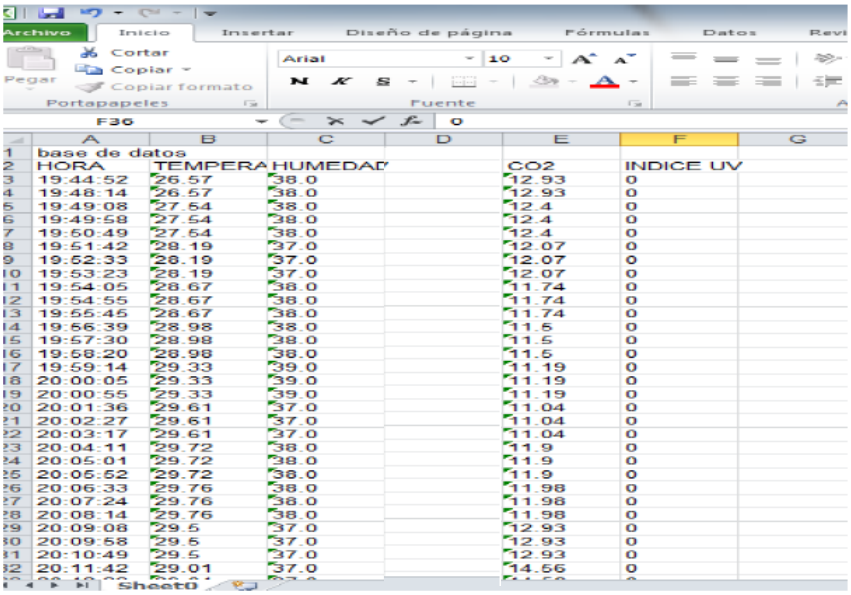


Figura 3.27 Base de datos en Excel

Tabla de mediciones de los parámetros ambientales.
Cabe mencionar que estas mediciones se realizaron en la intemperie dentro de su contenedor.

Tabla 3.2 parámetros ambientales de 12:21PM a 12:42PM

base de datos				
HORA	TEMPERATURA	HUMEDAD	CO2	INDICE UV
12:21:25	20.8	38	4.24	3
12:23:31	20.8	38	4.24	3
12:24:11	20.8	38	4.24	3
12:24:44	21.84	40	18.3	3
12:25:24	21.84	40	18.3	3
12:26:04	21.84	40	18.3	3
12:27:07	23.55	39	18.3	3
12:27:47	23.55	39	18.3	3
12:28:27	23.55	39	18.3	4
12:29:07	23.55	39	18.3	4
12:29:47	23.55	39	18.3	4
12:30:31	24.97	38	12.15	4
12:31:11	24.97	38	12.15	3
12:31:51	24.97	38	12.15	3
12:32:31	24.97	38	12.15	5
12:33:04	25.76	38	17.15	5
12:33:44	25.76	38	17.15	5
12:34:27	26.18	37	11.74	4
12:35:07	26.18	37	11.74	3
12:35:47	26.18	37	11.74	3
12:36:41	26.75	37	13.57	4
12:37:21	26.75	37	13.57	4
12:38:01	26.75	37	13.57	4
12:38:41	26.75	37	13.57	3
12:39:14	27.19	37	16.19	3
12:41:39	27.53	37	15.73	3
12:42:19	27.53	37	15.73	3

Tabla 3.3 parámetros ambientales de 16:15PM a 16:41PM

base de datos				
HORA	TEMPERATURA	HUMEDAD	CO2	INDICE UV
16:15:34	23.22	37.0	4.44	3
16:18:10	23.22	37.0	4.44	3
16:18:50	23.22	37.0	4.44	3
16:19:23	24.12	37.0	17.02	3
16:20:03	24.12	37.0	17.02	3
16:20:43	24.12	37.0	17.02	3
16:21:47	25.75	37.0	16.78	3
16:22:27	25.75	37.0	16.78	3
16:23:07	25.75	37.0	16.78	3
16:23:47	25.75	37.0	16.78	3
16:24:20	27.21	37.0	16.42	4
16:25:00	27.21	37.0	16.42	4
16:25:40	27.21	37.0	16.42	4
16:26:20	27.21	37.0	16.42	4
16:26:53	27.43	37.0	15.73	3
16:27:33	27.43	37.0	15.73	3
16:28:13	27.43	37.0	15.73	3
16:29:17	27.54	37.0	15.29	3
16:29:57	27.54	37.0	15.29	3
16:30:37	27.54	37.0	15.29	3
16:31:17	27.54	37.0	15.29	3
16:31:50	27.85	37.0	15.08	3
16:32:30	27.85	37.0	15.08	3
16:33:10	27.85	37.0	15.08	3
16:33:50	27.85	37.0	15.08	3
16:34:23	28.32	37.0	14.87	3
16:35:04	28.32	37.0	14.87	3
16:35:44	28.32	37.0	14.87	3
16:36:47	28.44	37.0	14.66	3
16:37:27	28.44	37.0	14.66	3
16:38:07	28.44	37.0	14.66	3
16:38:47	28.44	37.0	14.66	3
16:39:20	28.3	37.0	14.45	3
16:40:00	28.3	37.0	14.45	3
16:40:40	28.3	37.0	14.45	3
16:41:20	28.3	37.0	14.45	3
16:41:54	28.33	40.0	14.05	3

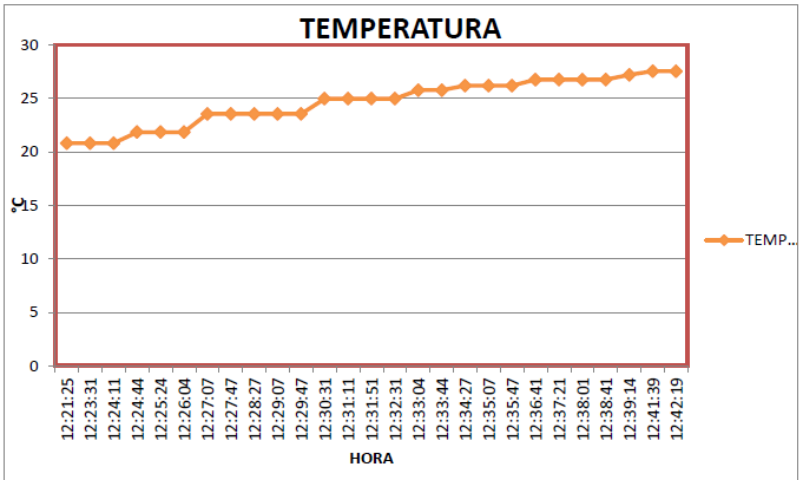


Figura 3.28 Grafica de temperatura [°C] cada 2:30 minutos

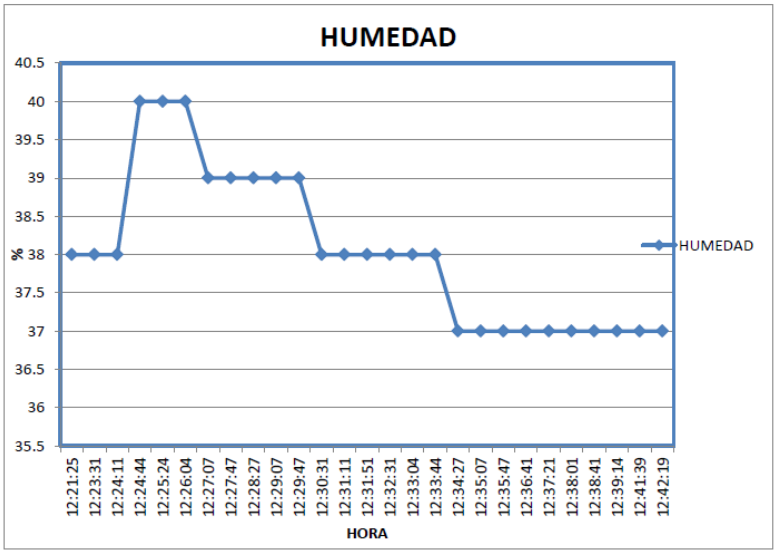


Figura 3.29 Grafica de Humedad relativa [%] cada 2:30 minutos

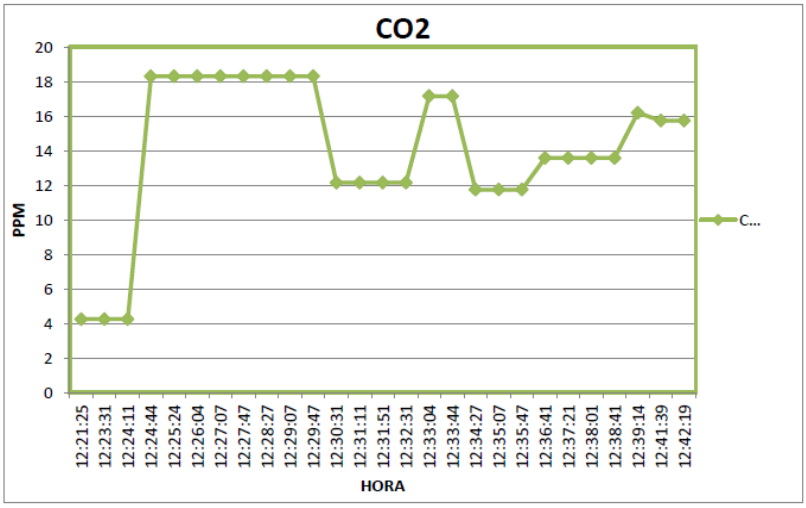


Figura 3.30 Grafica de Monóxido de carbono [PPM] cada 2:30 minutos

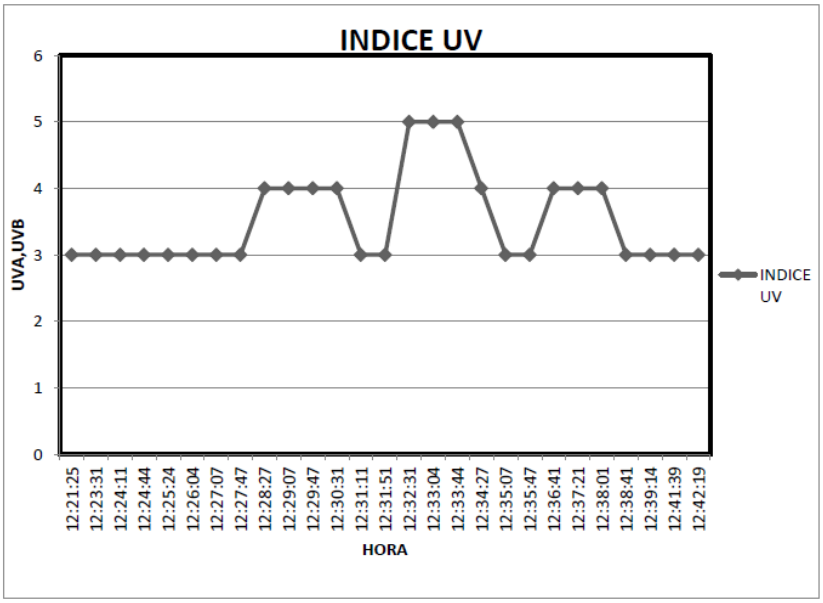


Figura 3.31: Grafica de Intensidad UV [UV] cada 2:30 minutos

3.6 OBSERVACIONES Y SUGERENCIAS.

Cabe mencionar que los ejemplos que el fabricante proporciona están configurados en una placa Arduino UNO, y nosotros utilizamos una placa Arduino Mega, pero no existe ningún inconveniente ya que por ser el mismo fabricante es exactamente la misma manera de configurar en las diferentes placas, tan solo en el compilador hay que especificar qué tipo de placa es la que estamos usando y proceder de la misma manera.

La configuración de pines en el Arduino solo varía al arreglo (acomodo) que se le fueron dando de la misma manera al introducir los códigos se declararon con el mismo número de pin al que estos estaban conectados.

En la cuestión de los sensores fue un poco complicado pero no imposible configurarlos ya que cuando los probamos en el laboratorio nos respondían de una manera casi exacta, pero al probarlo ya dentro del invernadero por razones de estar expuestos al ambiente estos entregaban mediciones alteradas, por lo que procedimos a corregir eso aunque no en su totalidad como hubiésemos querido, ya que el monitoreo entrega una muy buena lectura de los sensores pero con un rango de error de ± 5% lo cual debe ser considerado por los usuarios al hacer la lectura.

CONCLUSIÓN.

A través de la realización de este trabajo, se realizó una investigación de algunos parámetros ambientales en nuestro planeta, como también se ha diseñado e implementado una estación meteorológica a distancia. Su diseño se ha basado en conseguir medidas de parámetros ambientales como la temperatura, humedad, CO2 e intensidad UV mediante sensores y transmitir los datos por medio de tarjetas Arduino y un modem para acceder a una red, proponiendo así un nuevo sistema de medición alternativo al habitual por las estaciones meteorológicas comerciales, basado en entorno de programación web y almacenamiento de datos.

Se originó la idea de desarrollar un sistema de adquisición de datos portable y remota de bajo costo, esta razón aunada con los conocimientos que se adquirieron en la carrera de Ingeniería Electrónica, derivo que se lograra desarrollar este dispositivo funcional y de bajo costo.

Los sensores comerciales que se han utilizado han sido seleccionados siguiendo los siguientes criterios: disponibilidad del producto, coste y comportamiento físico.

En conclusión vemos la necesidad que tienen los invernaderos para la buena producción y conservación de las especies dentro de estos. Por lo que el monitorear los otros 3 invernaderos seria el siguiente paso ya que vimos que fue de mucha utilidad pero para solucionar el problema a mayor rango deberían monitorearse todos los invernaderos, y este prototipo de monitoreo puede ser utilizado no solo en los invernaderos del Instituto sino podrían mejorarse añadiendo la parte de control para que las correcciones al ambiente del invernadero sean automáticas, así no necesitaran de usuarios expertos en el tema de la agricultura para obtener una buena producción y conservación de los plantíos en los invernaderos.

REFERENCIAS

[1] http://html.rincondelvago.com/estacion-meteorologica.html

[2] http://platea.pntic.mec.es/~mhidalgo/documentos/02_PlataformaArduino.pdf

[3] http://arduino.cc/en/Main/ArduinoWirelessProtoShield

[4] Randy Russell, “Windows to the Universe: Radiation Ultraviolet (UV)”. National Earth Science Teachers Association ,2010.

[5] Solar Light Co., Inc., “A comparison of Spectroradiometers to Radiometers for UV Radiation Measurements”, págs. 35-36

[6] http://www.zigbee.org/

[7] http://www.digi.com/support/kbase/kbaseresultdetl.jsp?kb=184

[8] Fred Eady, Hands-on ZigBee, Elsevier, 2010

[9] Arduino and Kinect proyects, desing,build, blow their minds; autor Enrique Ramos Melgar and Ciriaco Castro Diez editorial Technology in action "TIA"

[10] Arduino programming notebook, Brian w. Evans, published: first edition august 2007 Second Edition September 2008

[11] Arduino + Ethernet Shield, “implantación de Arduino en las redes Ethernet: Arduino y el internet de las cosas”, serie: Arduino comunicación. José Manuel Ruiz Gutiérrez

[12] http://playground.arduino.cc/

ANEXOS

Figura 3.32 Armado del prototipo de estación meteorológica

Figura 3.33 Vista de la estación meteorológica de noche

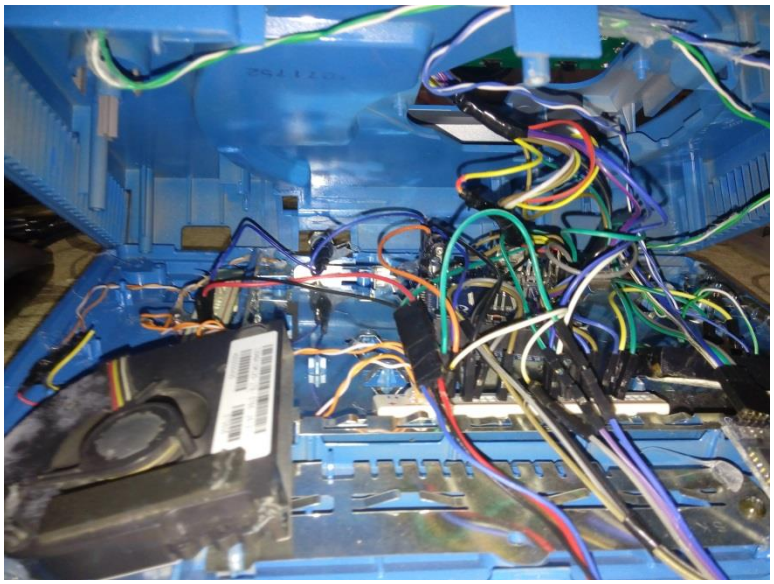


Figura 3.34 Vista del cableado interno

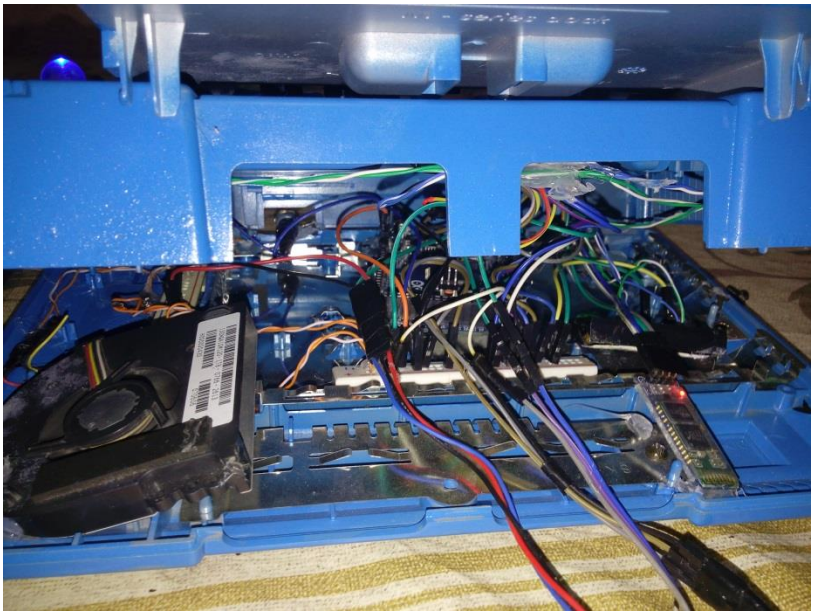


Figura 3.35 Vista del cableado y ventilador que mantiene una temperatura estable para las placas Arduino

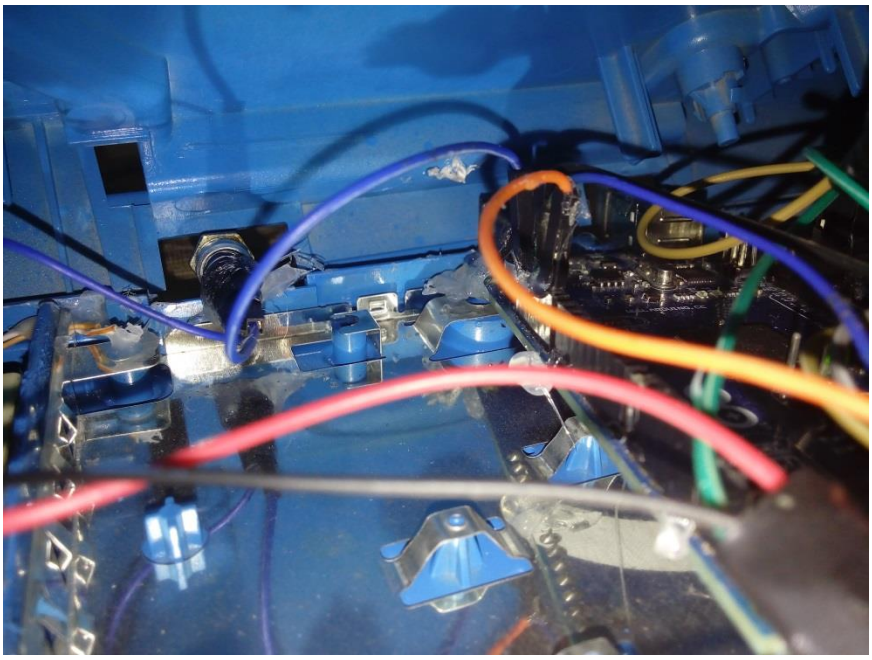


Figura 3.36 Arduino Mega en el interior de la estación meteorológica

Figura 3.37 Vista del invernadero en el que se implementó el proyecto

Figura 3.38 Conexiones de prueba dentro del invernadero

Figura 3.39 Colocación de sensores dentro del invernadero

Figura 3.40 Pruebas realizadas con presencia de compañeros Bioquímicos dentro del invernadero

Figura 3.41 Colocación de los sensores de humedad de suelo

Figura 3.42 Prueba de la estación meteorológica instalada en el invernadero

Figura 3.43 Estación meteorológica mostrando la humedad en el suelo

Figura 3.44 Estación meteorológica mostrando la temperatura y humedad en el aire