

Proyecto del curso Dispositivos Ubicuos

Memoria del proyecto

DISPOSITIVOS UBICUOS

Arturo Durán Domínguez

Por:

Daniel Correa Barrios



**Universidad de Extremadura
Escuela Politécnica
2015/16**

Contenido

1. [Contenido](#)
 - 1.1. [Elementos Utilizados](#)
 - 1.2. [Montaje](#)
2. [Funcionalidades Implementadas](#)
3. [Servicios de Terceros](#)
4. [Conclusiones](#)
5. [Referencias](#)

Introducción

En este documento se busca presentar el desarrollo del proyecto del curso Dispositivos Ubicuos. Se presentará en detalle el montaje y como esta se construyó el dispositivo además de la implementación del sketch y los scripts que corren sobre el Arduino Yun.

El objetivo del montaje es construir un sistema de monitoreo para una sala de cómputo, donde se busca conocer el valor de temperatura y humedad del lugar.

A continuación el desarrollo completo del proyecto:

Montaje y Construcción del Dispositivo

Elementos Utilizados

- **Arduino Yun**

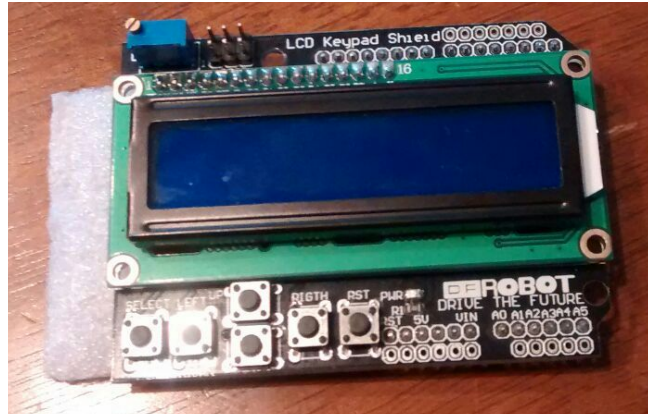


Necesario para la implementación del proyecto. En el se ejecuta el sketch y los scripts necesarios para hacer el monitoreo de temperatura y humedad de la sala de computo.

Cuenta con una tarjeta de red Wifi la cual permite tener acceso a internet para consultar los datos de temperatura y humedad por medio de una API REST, además de enviar la información recolectada a un servidor IoT para monitorear la sala desde cualquier lugar con acceso a internet.

Cuenta con un microprocesador y un sistema operativo sobre el cual se ejecutan los scripts necesarios para controlar la información, tener control de los dispositivos conectados y tener control interno del almacenamiento de la información.

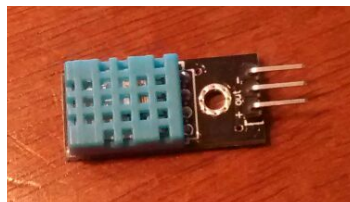
- **Pantalla LCD Keypad Shield**



Necesaria para mostrar los valores de temperatura, humedad y id del dispositivo que está capturando la información

Permite visualizar información relevante en tiempo real

- **Sensor DHT11**



Necesario para conocer la temperatura y humedad del lugar donde está ubicado.

Permite obtener los valores de temperatura en celsius y fahrenheit además del porcentaje de humedad del lugar

- Cables macho a hembra



Necesarios para conectar los dispositivos adicionales al Arduino Yun

Montaje

El montaje es la construcción del dispositivo. Para este caso particular el montaje es conectar la pantalla LCD y el sensor DHT11 al Arduino por medio de los cables macho a hembra, en las siguientes imágenes se muestra el dispositivo montado y funcionando.

Figura 1:

Montaje del Arduino con la pantalla LCD y el sensor DHT11 sin estar conectados al computador

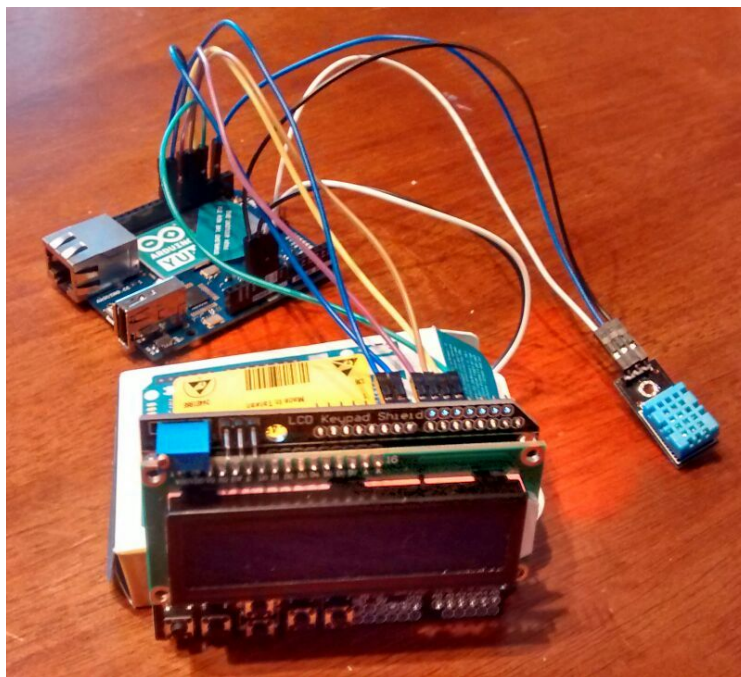


Figura 2:

Montaje completo conectado al computador y funcionando

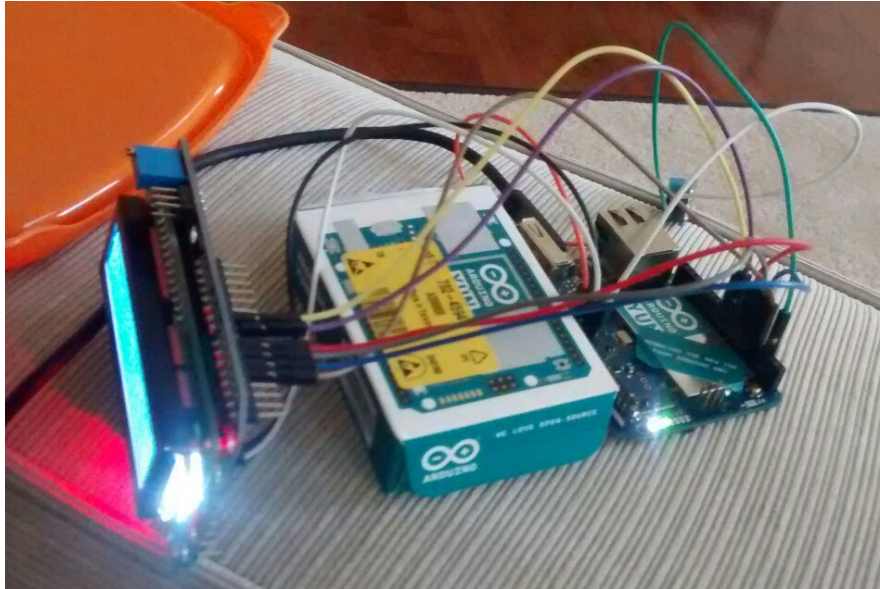


Figura 3:

Arduino conectado a la red Wifi y distribución de cables

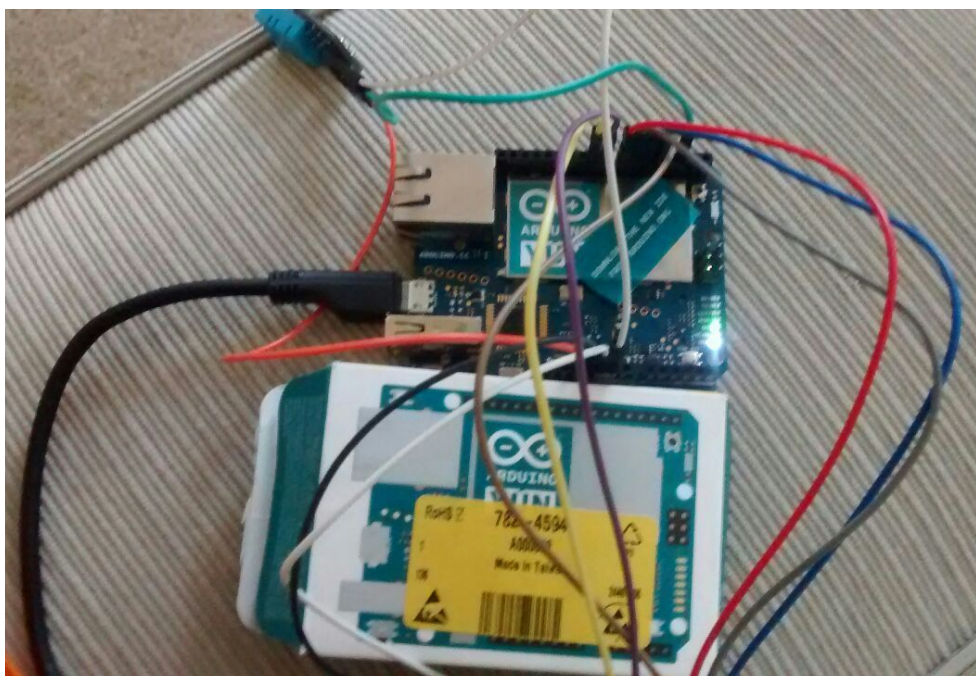


Figura 4:

Sensor DHT11 conectado al Arduino y recolectando datos

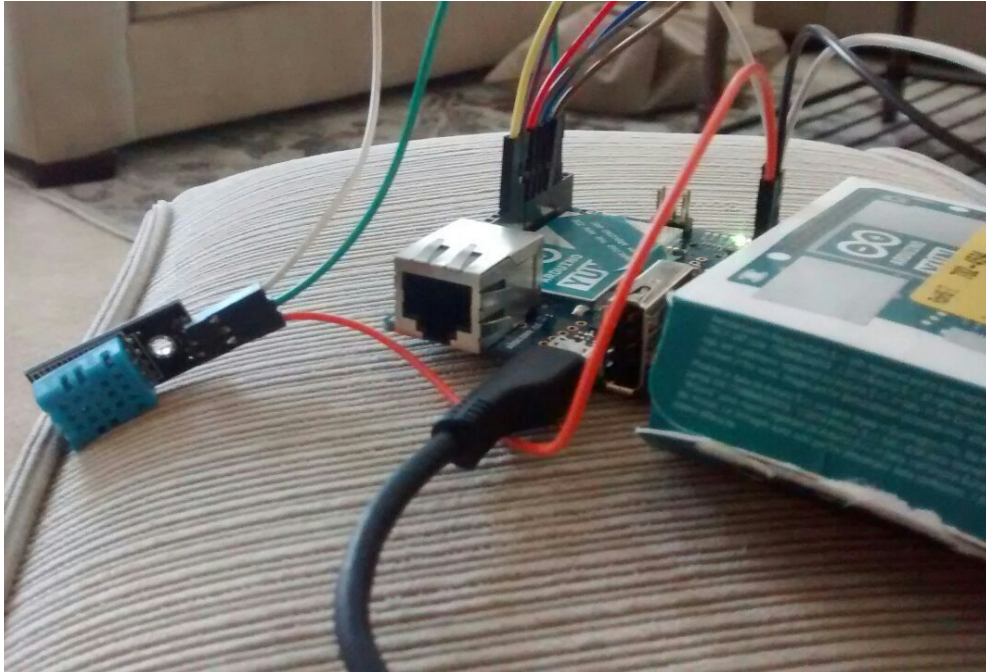
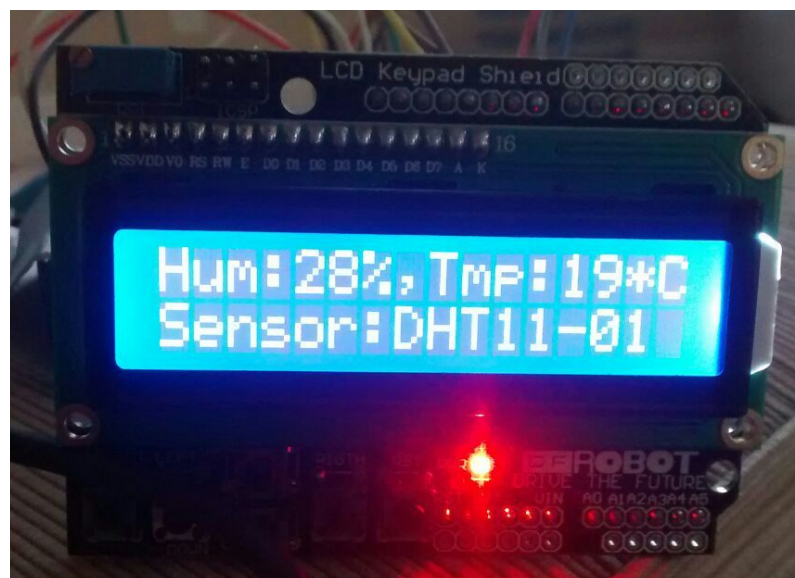


Figura 5:

Pantalla LCD mostrando los datos capturados



Funcionalidades Implementadas

Las funcionalidades principales se encuentran en el sketch que se ejecuta directamente en el sistema operativo del Arduino.

A continuación se detalla cada implementación realizada

Sketch

El sketch es el código fuente principal de la aplicación, se encarga de hacer la gestión del sensor, la pantalla LCD, la API REST, la escritura de archivos, control de fecha, control de errores y la ejecución del script que envía los datos al servidor en la nube. Los métodos que componen el sketch son:

- getHumedad(): Permite obtener el valor de humedad del sensor DHT11
- getTemperatura(): Permite obtener el valor de temperatura del sensor DHT11
- escribirLCD(int h, int t, String idSensor): Permite escribir los valores de humedad, temperatura e id del dispositivo en la pantalla LCD
- process_REST(YunClient client, int h, int t): Se encarga de procesar la petición realizada por medio de la API REST
- iniciarAPIREST(int h, int t): Se encarga de verificar la existencia de peticiones por medio de la API REST
- isError(int h, int t): Se encarga de verificar la existencia de errores en la lectura del sensor y actuar frente a errores
- getDate(): Se encarga de obtener la fecha
- escribirInformacion(String id, int h, int t): Se encarga de escribir la información recolectada en el archivo log
- envioDatosServer(): Se encarga de ejecutar el script Python que envía los datos al servidor en la nube
- setup(): Se encarga de inicializar variables y realizar configuración inicial
- loop(): Método principal, ejecuta en orden los métodos necesarios en cada iteración. Controla el tiempo de ejecución y los tiempos de ejecución del script de envío de datos al servidor

El código fuente se encuentra en el archivo sketch.ino mientras que el archivo hex para la ejecución en el sistema operativo del Arduino se llama sketch.hex

Sensor DHT11

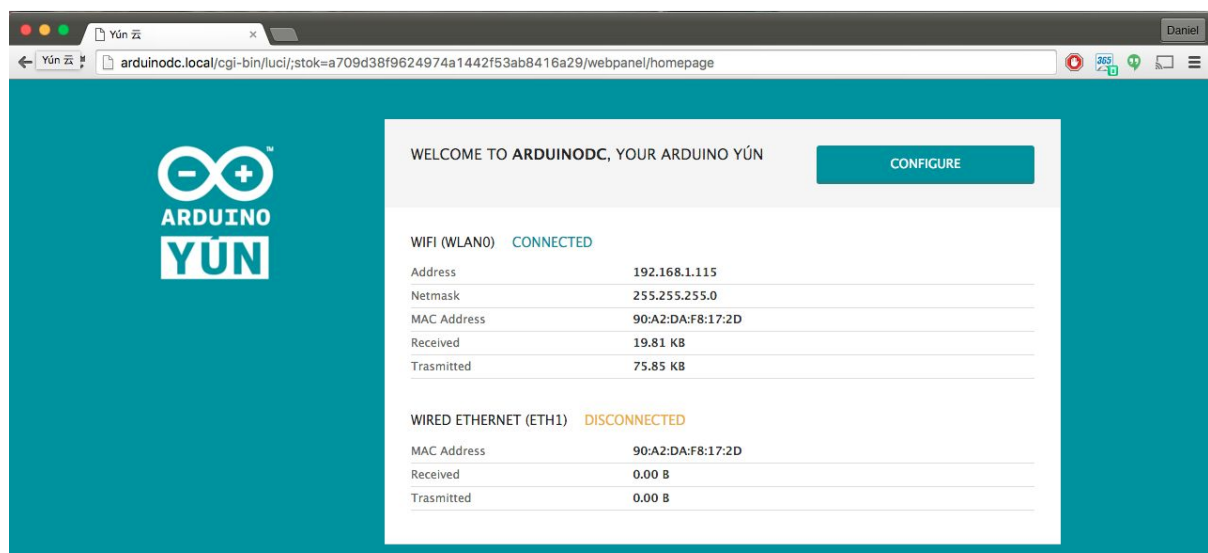
Para recibir la información del sensor se utilizó el pin digital número 2 del Arduino. Por medio de la librería del sensor se leen los datos de temperatura y humedad en el sketch. La implementación se encuentra en los métodos getHumedad() y getTemperatura().

Conexión a internet

Para poner en funcionamiento la API REST y el envío de datos al servidor IoT fue necesario conectar el Arduino a internet por medio de la tarjeta de red Wifi.

Se configuró la dirección del arduino como arduinodc.local, por medio de esta dirección se puede acceder al Arduino mediante un navegador web con HTTP o por medio de una terminal con SSH.

La información de conexión del Arduino y su configuración se puede gestionar mediante el navegador web. Cuando el Arduino no está conectado a una red permite conectarse a su propia red, dentro de esta accedemos a su dirección ip (192.168.240.1) en el navegador por el puerto 80 y procedemos a realizar la configuración.



API Rest

La API Rest fue implementada en el sketch principal en los métodos iniciarAPIREST(int h, int t) y process_REST(YunClient client,int h, int t). Para acceder a la API REST se debe ingresar a la dirección arduinodc.local/arduino/temp o arduinodc.local/arduino/hum donde se pueden consultar los valores de temperatura y humedad respectivamente. Esta API se implementó con la librería YunServer y YunClient para la creación del cliente y servidor encargados de realizar y responder la petición respectivamente

A continuación se adjuntan los pantallazos con las consultas realizadas

Temperatura:



Humedad:



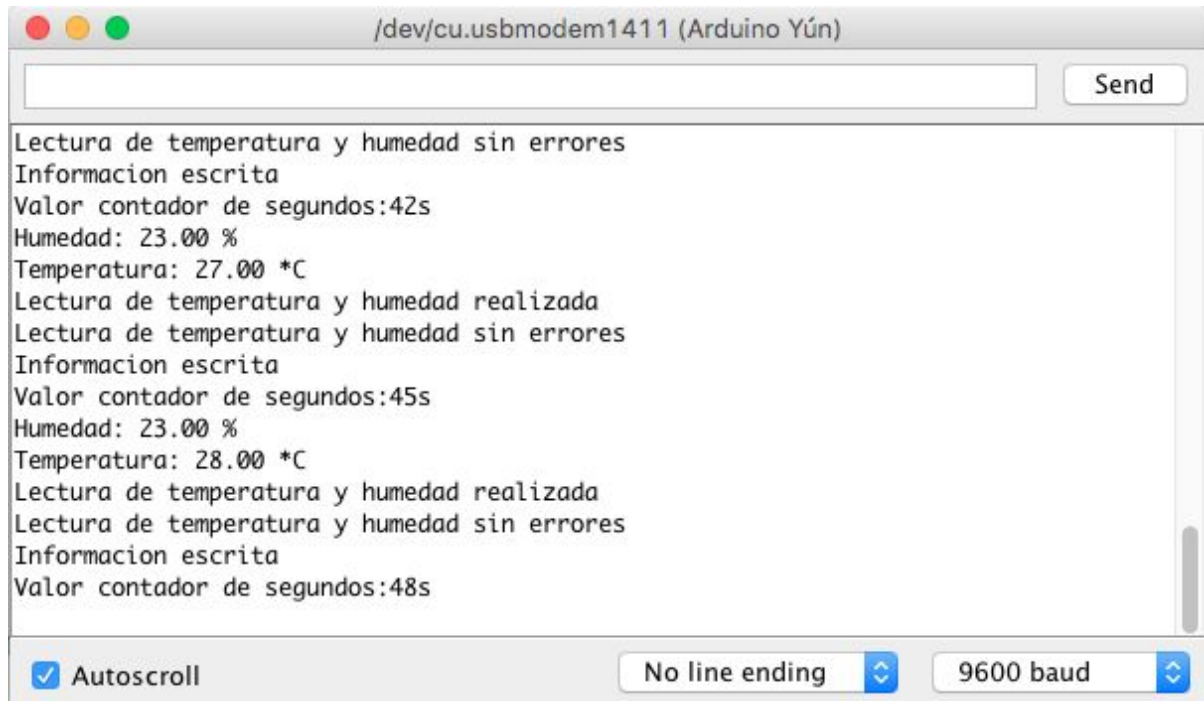
Pantalla LCD

La visualización de temperatura, humedad e id del dispositivo en la pantalla LCD se implementó en el sketch principal, la pantalla está conectada a la entrada de 5v y gnd y para visualización a los pines digitales 8, 9, 4, 5, 6, 7

La implementación está en el método escribirLCD(int h, int t, String idSensor) y utiliza la librería LiquidCrystal facilitada por el fabricante de la pantalla

Monitor Serie

Se utiliza el monitor serie para imprimir información de lectura, ejecución y errores. Se busca tener un seguimiento completo de la aplicación por lo que se informa cada proceso que se realizó correctamente o si hubo algún error. A continuación una captura de pantalla del monitor serie en ejecución:



```
Lectura de temperatura y humedad sin errores
Informacion escrita
Valor contador de segundos:42s
Humedad: 23.00 %
Temperatura: 27.00 *C
Lectura de temperatura y humedad realizada
Lectura de temperatura y humedad sin errores
Informacion escrita
Valor contador de segundos:45s
Humedad: 23.00 %
Temperatura: 28.00 *C
Lectura de temperatura y humedad realizada
Lectura de temperatura y humedad sin errores
Informacion escrita
Valor contador de segundos:48s
```

☒ Autoscroll No line ending 9600 baud

Logs

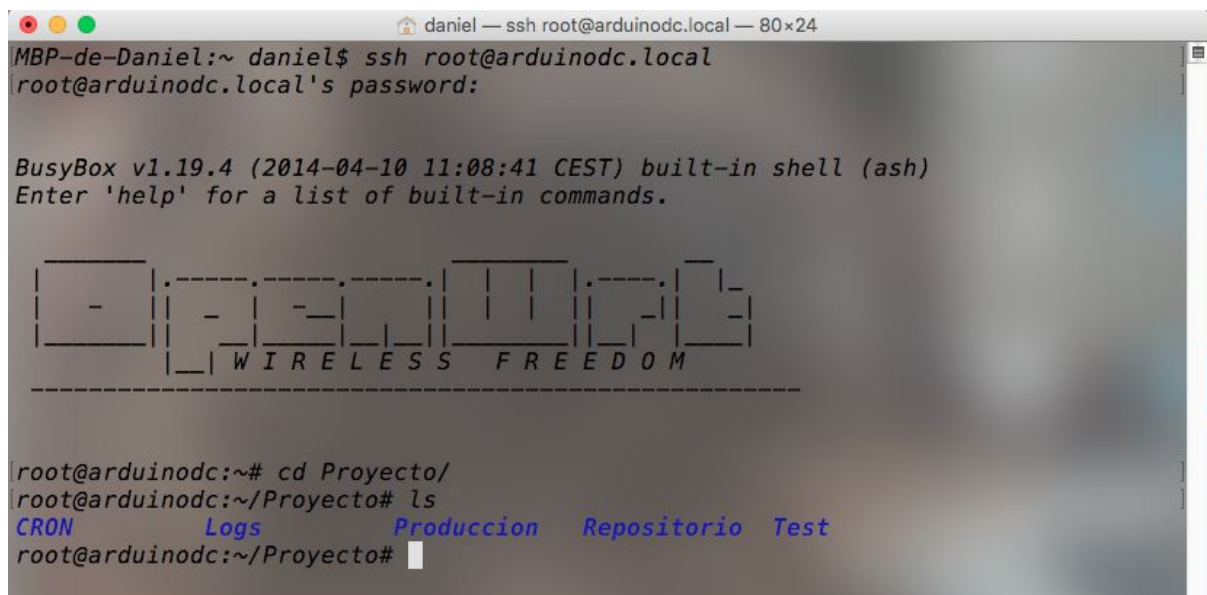
La escritura de logs esta implementada en el sketch principal particularmente en el método `escribirInformacion(String id,int h, int t)`. Con la librería `FileIO` se accede (o se crea) el archivo `sensor.log` en el cual se escriben los valores de temperatura, humedad, fecha e id del dispositivo en el formato solicitado en el proyecto.

Estos logs son almacenados en el director `Logs`, en el siguiente ítem se explica con mas detalle la jerarquía del proyecto.

Jerarquía del arduino

Para la gestión del proyecto dentro del sistema operativo del Arduino fue necesario definir una jerarquía y orden para los directorios que contienen los scripts, el `.hex`, los logs ademas de los test y el repositorio.

El orden definido fue el siguiente:



- El directorio CRON contiene el script que se ejecuta cada vez que se inicia el Arduino
- El directorio Logs contiene el archivo sensor.log, un directorio con los logs antiguos y otro con el .zip de los logs comprimidos
- El directorio Produccion contiene el .hex mas actualizado y un directorio con los scripts para el envío de datos al servidor
- El directorio Repositorio contiene el .hex más reciente subido por medio de ssh (scp), en este directorio se sitúa cualquier actualización realizada al sketch del Arduino.
- El directorio Test contiene archivos de prueba como .logs y .hex

Script de actualizaciones y gestión de logs

Para la actualización del archivo .hex del repositorio al directorio de producción se implementó un script en python llamado `cron_arduino.py`. Este archivo está almacenado en el directorio CRON y se ejecuta cada vez que el arduino se inicia. Este script tiene un método (`modification_date(path)`) para consultar la fecha de modificación del archivo ubicado en `path` y el método principal (`cron_arduino()`) para verificar las fechas de modificación, intercambiar los archivos y ejecutar las líneas de comando que suben el .hex al linino.

Por otro lado para la gestión de logs se tienen varios scripts escritos en python:

- `zip_logs`: Comprímé los logs ubicado en `old_logs` eliminandolos del directorio y dejándolos comprimidos en un `.zip` ubicado en `zip_logs` con la fecha de compresión
- `move_logs`: mueve el archivo `sensor.log` al directorio `old_logs` agregando a su nombre la fecha que se mueve.

Estos scripts se encargan de hacer la gestión de los logs requerida en el proyecto

CRON y ejecuciones en el linino

Para realizar la gestión es necesario configurar en el sistema operativo los cron de ejecución de cada script ya que los instantes de ejecución de cada uno son diferentes.

La configuración propuesta es la siguiente:

- Borrado de logs cada 3 meses:

```
0 0 1 4,7,10,1 * rm /root/Proyecto/Logs/zip_logs/*
```

- Mover el archivo sensor.log cada semana:

```
0 0 * * 1 python /root/Proyecto/Produccion/scripts/move_logs.py
```

- Comprimir logs cada 2a semana del 3er mes:

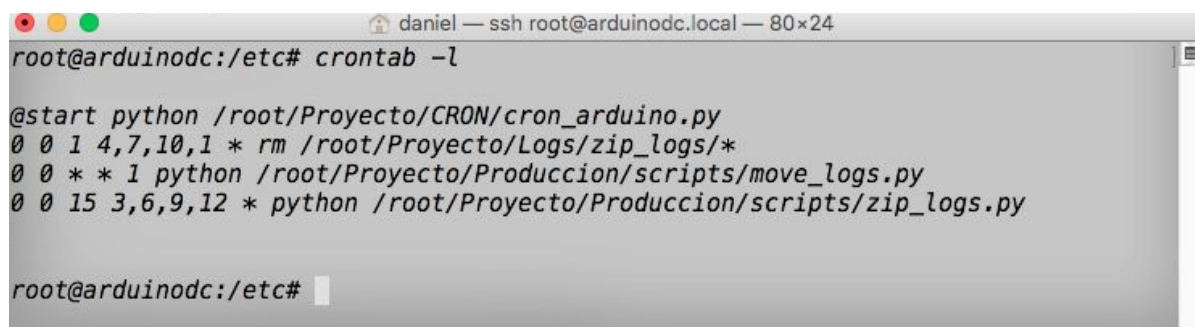
```
0 0 15 3,6,9,12 * python /root/Proyecto/Produccion/scripts/zip_logs.py
```

Para realizar esta configuración en un crontab se ejecuto el comando *crontab -e* en el inino
Para el script que debe ejecutarse cada que vez que inicie el arduino se agrego la siguiente línea en el archivo rc.local ubicado en el directorio /etc

```
python /root/Proyecto/CRON/cron_arduino.py
```

Esta configuración se encuentra en el archivo cron_order.txt y fue escrita en el crontab del arduino

La configuración del crontab se puede ver en la siguiente captura:



```
daniel — ssh root@arduino.local — 80x24
root@arduino.local:/etc# crontab -l

@start python /root/Proyecto/CRON/cron_arduino.py
0 0 1 4,7,10,1 * rm /root/Proyecto/Logs/zip_logs/*
0 0 * * 1 python /root/Proyecto/Produccion/scripts/move_logs.py
0 0 15 3,6,9,12 * python /root/Proyecto/Produccion/scripts/zip_logs.py

root@arduino.local:/etc#
```

Sistema de monitorización en la nube

Para el envío de datos a la nube se utilizó el servicio initial state (<https://www.initialstate.com>) el cual permite tener gráficas de las últimas 24 horas de datos enviados a su servidor. Se explicará más en detalle su software en la sección servicios de terceros

Script para envío de datos y script de creación de buckets

El envío de datos a la nube se implementó en un script aparte del sketch.ino, sin embargo, el sketch es el encargado de manejar los tiempos y realizar su ejecución.

Dentro del .ino se lleva el conteo de tiempo de ejecución hasta completar 3 minutos, una vez se cumplan los 3 minutos se ejecuta el script encargado del envío y el contador es cambiado a 0

El primer script (post_bucket.py) permite crear el bucket principal para el envío de información al servidor.

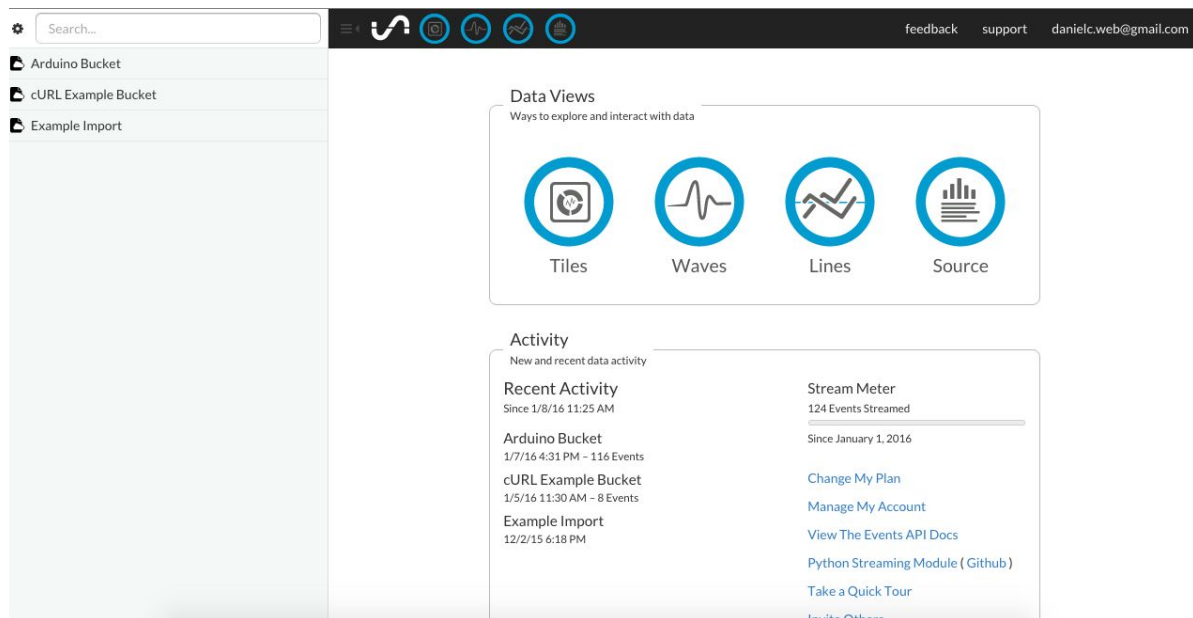
El segundo script (send_data.py) recupera la última información escrita en el log y la envía al servidor mediante una petición POST en un archivo JSON utilizando curl. En la siguiente sección se explica porque se eligió curl como estrategia para el envío de datos.

Servicios de Terceros

InitialState

InitialState es un servicio en la nube que permite visualizar información enviada a sus servidores en diferentes tipos de visualización. Tienen soporte para distintos lenguajes de programación, dispositivos y servicios web. Su servicio gratuito permite enviar 25K eventos por mes y visualizar información de las últimas 24 horas. En sus planes de pago se permite mas envio de datos y visualización completa de la información.

Para poder utilizar el servicio es necesario crear una cuenta y seleccionar un plan, una vez registrado tenemos un panel de visualización y un administrador de cuenta:



En el panel de visualización se pueden seleccionar los buckets creados y las visualizaciones disponibles además de información básica del número de eventos utilizados y actividad reciente.

Streaming Access Keys

g7ewgXN0eJpaizQaCLJ7kwtswBNKmU19
Issued 12/2/15 6:19 PM - [Remove Key](#)

[+ Create A New Key](#)

Streaming Connections



En el panel de administración se pueden gestionar las Access Keys para el envío de datos.

Un bucket es una sección a la cual enviamos datos, cada bucket es independiente y puede contener información diferente. Se utilizan para diferenciar dispositivos o conjuntos de datos. Cada API tiene su soporte para creación de buckets

Bucket de prueba

Cada proyecto viene con un bucket de prueba para aprender a manejar las visualizaciones. Este bucket se llama Example Import y contiene información proporcionada por InitialState.

Para las pruebas de envío iniciales se creó el bucket cURL Example Bucket el cual fue utilizado para probar el script con información redundante.

Bucket del proyecto

Para el desarrollo del proyecto se utilizó el bucket Arduino Bucket, este bucket es creado mediante el script `post_bucket.py` el cual envía una petición POST al servidor con la información del bucket para la creación del mismo.

En este bucket se encuentra la información del proyecto, es utilizado por el script de envío de datos y permite visualizar los datos de las últimas 24 horas.

API con RESTful

InitialState ofrece una API de envío de datos mediante el protocolo REST. Mediante una petición POST enviando un archivo en formato json el servidor puede almacenar y visualizar la información enviada.

El ejemplo de envío de datos con REST está en el enlace: <http://support.initialstate.com/knowledgebase/articles/590091-how-to-stream-events-via-restful-api>

API con Arduino

InitialState ofrece una API para trabajar directamente con el Arduino facilitando un ejemplo de envío de datos desde un sketch.

El proyecto completo de envío de datos del arduino se encuentra en el siguiente enlace: <https://github.com/InitialState/arduinosenorbox/wiki>

API con Python

InitialState ofrece una librería concreta para envío de datos desde Python, esta librería facilita la creación de peticiones POST ya que permite trabajar el envío como si fueran logs.

El ejemplo de envío de datos con Python se encuentra en el siguiente enlace: <http://support.initialstate.com/knowledgebase/articles/590085-how-to-stream-events-in-python>

Elección de API de envío

Se decidió trabajar y hacer el envío en Python utilizando la API RESTful por las siguientes razones:

- No fue posible instalar la librería que provee InitialState para trabajar con Python en el Arduino, motivo por el cual fue descartada. Además, su uso podría suponer un

peso adicional a la carga del script, costo que no es posible asumir en esta clase de proyectos.

- El sketch del proyecto ya era lo bastante robusto como para encargarle la tarea del envío de información al servidor. La responsabilidad se dividió de tal manera que el sketch hace la gestión del instante de ejecución y realizará la ejecución del script encargado de enviar la petición con los datos al servidor
- La API REST que provee InitialState es la manera más eficiente de intercambio de datos ya que es independiente de tecnologías y se puede utilizar desde la implementación que se desee.

Conclusiones

El desarrollo del proyecto ha sido un gran aporte al aprendizaje de la tecnología Arduino. En el avance del curso he podido trabajar con diferentes dispositivos que integrados pueden llegar a cumplir objetivos interesantes. La propuesta implementada puede ser útil en salas de cómputo como fue propuesta inicialmente pero podría ser útil para realizar monitoreo de temperatura y humedad en muchos otros contextos.

Al inicio del curso no tenía experiencia alguna en trabajo directo con hardware y en especial con Arduino, el desarrollo del curso me ha enseñado a manipular y trabajar con estos dispositivos al nivel de pensar en ideas que podrían implementarse fácilmente con un dispositivo de esta clase.

Ha sido muy productivo y satisfactorio el trabajo realizado, la metodología utilizada estimula la creatividad y crea un ambiente divertido, ya que compartir el desarrollo con más personas y trabajar en la práctica junto a la teoría permite interesarse mucho más en el aprendizaje de cualquier tecnología.

La experiencia fue muy gratificante y espero seguir aprendiendo mas al respecto, me queda como enseñanza el trabajo y como motivación el seguir aprendiendo más. Me faltó un poco más de tiempo para seguir trabajando en los requerimientos adicionales del proyecto.

Referencias

- I. Initial State Works With." 2015. 8 Jan. 2016 <<https://www.initialstate.com/workswith>>
- II. Data Streaming Blog | Initial State." 2014. 8 Jan. 2016 <<http://blog.initialstate.com/>>
- III. Initial State Support Documentation." 2015. 8 Jan. 2016 <<http://support.initialstate.com/>>
- IV. "Crontab – Quick Reference | Admin's Choice - Choice of ..." 2010. 8 Jan. 2016 <<http://www.adminschoice.com/crontab-quick-reference>>
- V. Arduino - Home." 2015. 8 Jan. 2016 <<https://www.arduino.cc/>>
- VI. Material de clase del curso Dispositivos Ubicuos, Universidad de Extremadura, 2015