

# **PROYECTO FINAL DEL CURSO IoT - RASPBERRY PI (fse23)**

## **AGRICULTURA 4.0 CON IoT Y RASPBERRY P**

**CIFO La Violeta**

**Realizado por:**

**Antonio Rizzo**

**Noviembre 2022**

## INTRODUCCIÓN

La población mundial aumenta todos los días y con ella la demanda de alimentos. Los métodos tradicionales de cultivo se basan en realizar determinadas acciones basados principalmente en el tipo de cultivo y de terreno, el conocimiento del agricultor y en las épocas del año. Este sistema, a pesar de ser exitoso, ya que ha provisto a la humanidad de alimentos a lo largo de toda la historia, consume muchos recursos, pesticidas, fertilizantes y principalmente agua que escasea cada día más.

Es necesario establecer un balance entre la producción agrícola y la utilización óptima de los recursos de manera de lograr una producción sustentable.

Los agricultores carecen de herramientas para recolectar información que sustente la toma de decisiones para mejorar su producción y hacer un uso mas racional de esos recursos.

La revolución agrícola 4.0, también conocida como agricultura 4.0 o como “SmartFarming” (Banco Interamericano de Desarrollo, 2019), busca introducir tecnologías actuales como la Robótica y Automatización, Internet de las Cosas, modelos predictivos con Inteligencia Artificial en los procesos productivos agrícolas, de manera de utilizar la cantidad justa de agua, fertilizantes y fumigación, en el momento justo y sólo en el lugar donde se requieren. A este proceso también se lo ha denominado como Agricultura de Precisión.

## OBJETIVO

Este trabajo tiene como objetivo explorar la factibilidad de construir un prototipo de “Granjas Inteligentes” en un invernadero con una Raspberry Pi que desempeñará el rol de Coordinador, un Arduino que hará de Sensor y Actuador

, los dispositivos (sensores y actuadores) y los conocimientos obtenidos en el curso de IoT y Raspberry.

El proyecto se desarrollará en tres etapas:

### ETAPA 1:

Establecer la comunicación entre el equipo que realiza la recolección de la data de los sensores (la información la suministran los diferentes sensores en los equipos) y el coordinador/concentrador que será una Raspberry Pi.

### ETAPA 2:

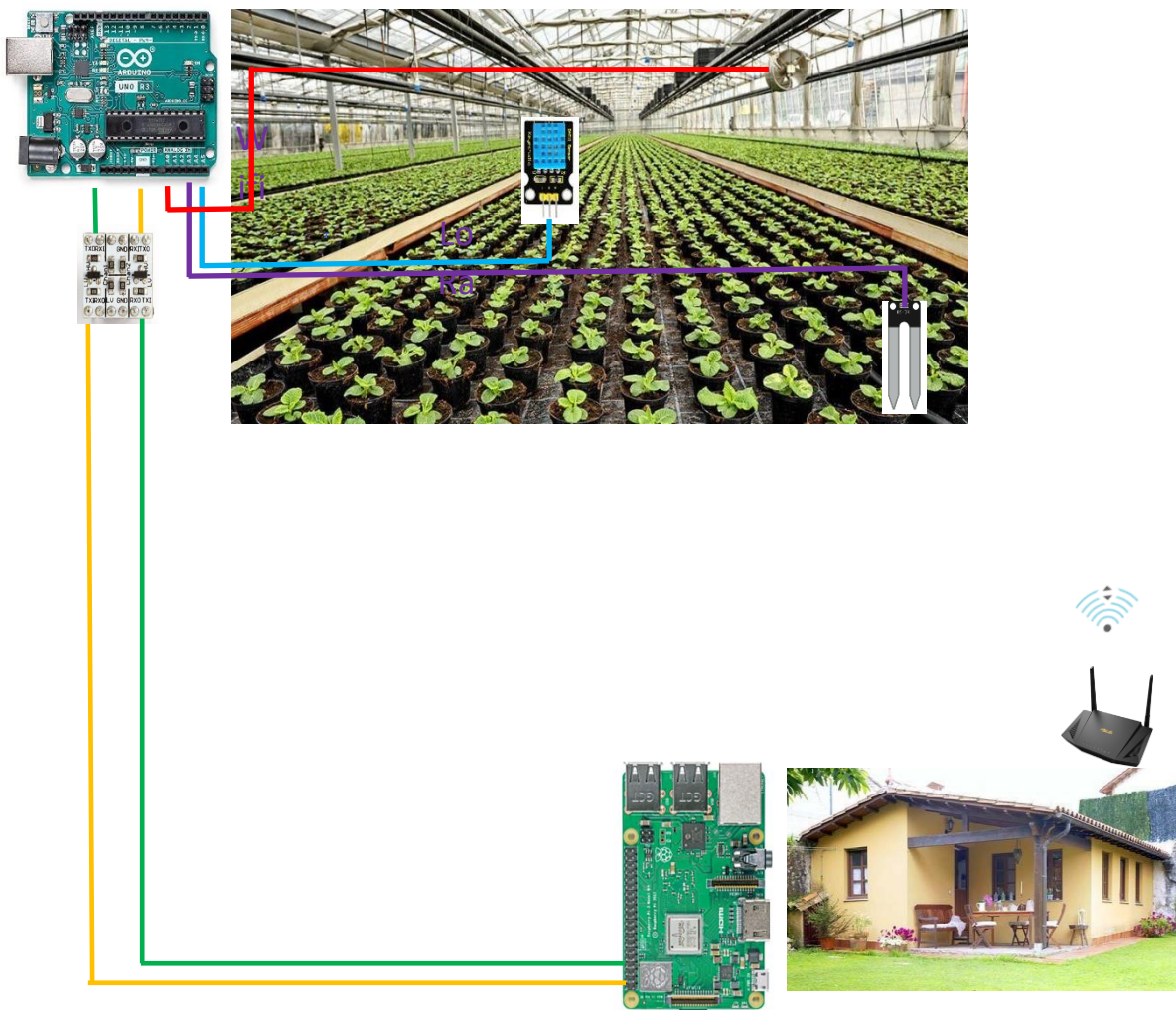
Lograr el envío de la información recolectada de los sensores a la nube. Esta información será la necesaria para alimentar el modelo de inteligencia artificial que realizará las predicciones y determinará las necesidades de Riego, Fertilización, Fumigación y Ventilación.

### ETALA 3:

El nodo Coordinador recibe los resultados del modelo predictivo y envia al nodo Actuador mensajes específicos. Este a su vez, decodifica el mensaje y genera las ordenes para los actuadores correspondientes.

En forma gráfica el sistema que se desarrollará será el siguiente.

PROPUESTA Provento AGRICULTURA 4.0 CON IoT Y RASPBERRY PI



## **ETAPA 1**

### **COMUNICACIONES SERIALES EN IoT**

Las comunicaciones seriales son un método utilizado para transmitir y recibir información un bit a la vez. Existen varios protocolos de comunicaciones seriales que en su conjunto se conocen como USART por sus siglas en ingles Universal Synchronous / Asynchronous Receiver Transmitter. Los protocolos síncronos requieren que el emisor envíe una señal de reloj al receptor para que se sincronicen, lo cual no es necesario en comunicaciones asíncronas esto. Los más conocidos son:

SPI: Serial Peripheral Interface

I2C: Inter Integrated Circuit

USB: Universal Serial Bus

UART: Universal Asynchronous Receiver Transmitter

Los protocolos I2C y SPI son protocolos síncronos, que funcionan con una arquitectura maestro-esclavo. En esta arquitectura existen dos tipos de dispositivos:

- El Maestro (uno en el caso de SPI y varios en el caso de I2C) que es el dispositivo que inicia y coordina la comunicación y quien envía la señal de reloj para la sincronización. En IoT este papel lo desempeñan las Raspberry Pi y/o los Arduinos
- Los Esclavos o Periféricos que son los dispositivos que están a la espera de que algún maestro se comunique con ellos. En este grupo encontramos a los sensores y los actuadores.

Los protocolos I2C y SPI permiten velocidades de transferencia de datos mucho más rápidas, del orden de los megabits, y están pensados para comunicaciones entre componentes de una misma placa de circuito, mientras que las comunicaciones asíncronas permiten la transmisión de datos a velocidades del orden de los kilobits pero a distancias de hasta unos 15 metros y con menos errores.

La comunicación serial asíncrona (UART), se lleva a cabo entre dos dispositivos, ninguno de los cuales es maestro o esclavo. Este protocolo es importante porque muchos microcontroladores, periféricos y sensores vienen preparados para comunicarse de esta manera. Como dijimos, no requiere señal de reloj, por lo que utiliza un cable menos que las señales síncronas, es más confiable y permite enviar datos a mayor distancia.

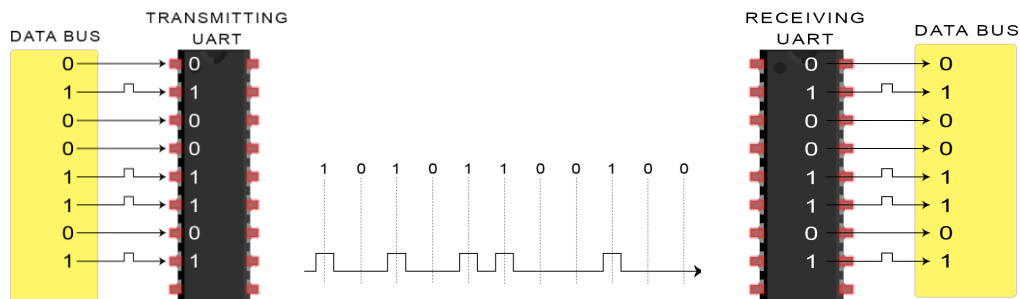
Al no tener señal de reloj, requiere que ambos dispositivos estén configurados para trabajar a la misma velocidad de transmisión y recepción de datos (baud rate).

Por lo mencionado anteriormente, para este proyecto comunicaremos una Raspberry Pi con un Arduino utilizando el protocolo UART, ya que requerimos que cualquiera de los dos pueda iniciar la comunicación y que ambos equipos no estarán en el mismo lugar.

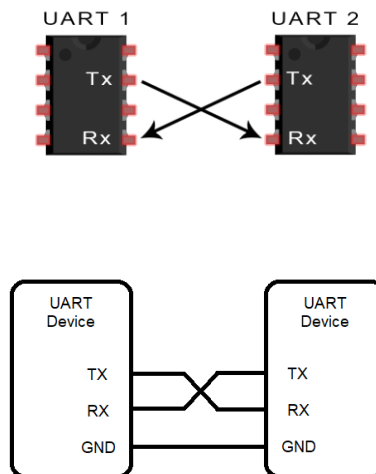
Explicaremos entonces con un poco mas de detalle, en qué consiste la Comunicación Serial Asíncrona o UART.

El puerto UART es un componente de hardware, bien sea integrado dentro del microcontrolador o en un chip externo; sus funciones son: tomar la data de un dispositivo en forma paralela y

convertirla a serial para su transmisión y recibir la data en forma serial y convertirla de regreso a paralela para entregarla al dispositivo. Al modular la señal del pin Tx, poniéndola en Alto o HIGH para indicar un uno(1) y en Bajo o LOW para indicar un cero(0), se transmite el mensaje en forma binaria, un bit tras de otro, como se indica en la siguiente figura.

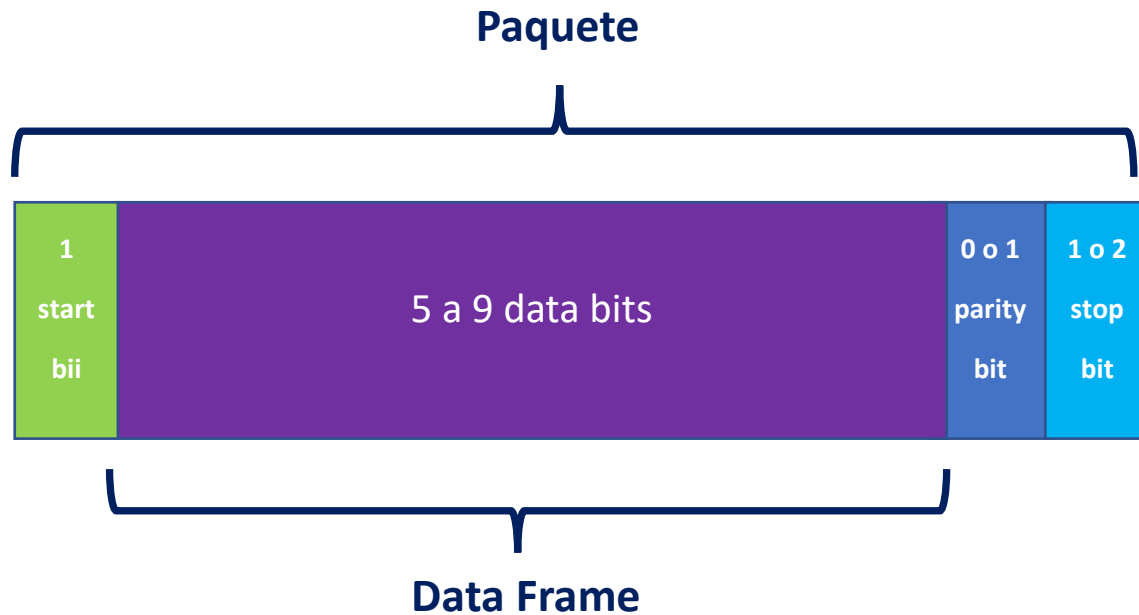


En cada uno de los dispositivos que se comunicarán, en nuestro caso una Raspberry y un Arduino, el puerto serial UART esta conectado a dos pines: un pin de Transmisión, designado con las siglas Tx y un pin de Recepción, que se designa con las siglas Rx. El pin Tx del dispositivo que transmita, se conecta mediante un cable al pin Rx del otro dispositivo y viceversa como se indica en la figura.



De manera simple, el protocolo Serial funciona así: el pin Tx del transmisor permanece en HIGH cuando no está transmitiendo y cuando tiene algo que transmitir baja el pin a LOW (manda un bit en 0) que le indica al receptor que ve el cero en su pin Rx, “voy a enviar un mensaje”. A este bit se le denomina Start Bit. Seguidamente envía un byte (ocho bits) de información, denominado Data Frame. Después del Data Frame se envían un bit de paridad ,opcional, para verificar la integridad de la data recibida y un último bit, denominado Stop Bit, que siempre es HIGH y que sirve para preparar la línea para la siguiente transmisión, la cual, tal como dijimos, comenzará poniendo la

línea en LOW. Al conjunto de todos los bits enviados se le denomina Paquete y su tamaño varía entre 7 y 12 bits siendo 10 bits el valor más usado.



Para que el mensaje sea leído correctamente, ambos dispositivos deben operar a la misma frecuencia de modulación o la misma tasa de bits por segundo. A esta tasa se le denomina baud rate y puede variar entre 9600 baudios y 115.200 baudios. Además en ambos dispositivos se deben configurar los mismos parámetros del paquete.

El proceso es como sigue: Una vez que el receptor detecta el Start Bit, espera 1,5 veces el tiempo de muestreo, que en el caso de una transmisión a 9600 baudios el tiempo de muestreo es  $1/9600 = 0,00010416$  segundos para asegurarse de “leer” justo en la mitad del bit y así reducir los errores por pequeñas desviaciones que pudiesen existir entre los relojes de ambos dispositivos, y continúa “leyendo” un bit tras otro hasta terminar de “leer” todo el Paquete acordado y espera hasta el próximo Start Bit, cuando se repite el proceso. Para calcular la velocidad de transmisión de la información, dividimos la velocidad de transmisión de bits (baud rate) entre el tamaño del paquete. Por ejemplo, en una configuración donde se transmitirá un byte a la vez, tendremos:

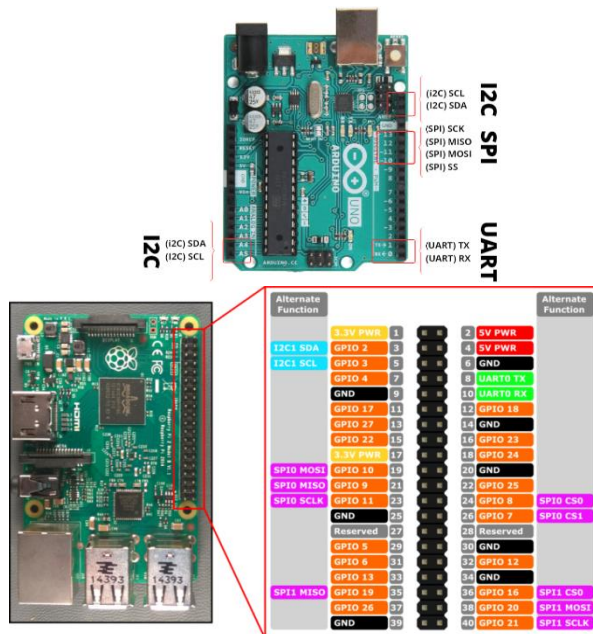
$$1 \text{ bit (Start Bit)} + 8 \text{ bits (Data Frame)} + 1 \text{ bit (Stop Bit)} = 10 \text{ bits}$$

$$9600 \text{ baudios (bits por Segundo)} / 10 \text{ bits} = 960 \text{ caracteres por segundo}$$

(aproximadamente 10 líneas de texto por segundo)

## CONFIGURACIÓN DE HARDWARE PARA LA COMUNICACIÓN SERIAL

Como parte de este proyecto, configuramos una Raspberry Pi 3 y un Arduino Uno para que se comuniquen. A continuación se muestran los distintos puertos de comunicación que poseen ambas placas, cada uno de los cuales tiene su respectiva librería.

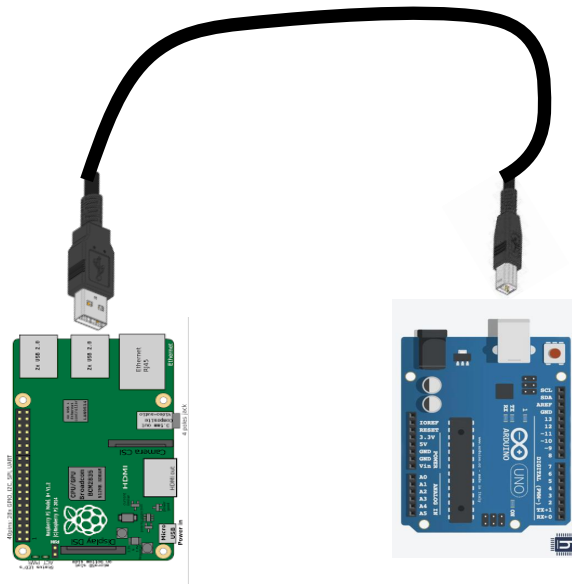


Existen dos formas para conectar la Raspberry Pi con el Arduino para establecer una comunicación Serial (UART) entre ambos: Vía puertos USB o a través de los pines Tx/Rx.

### Conexión vía Cable USB

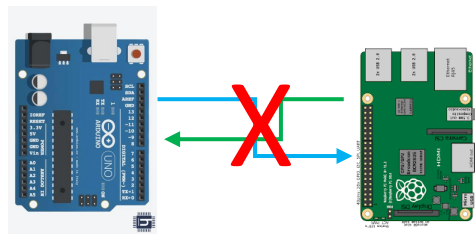
Esta es la manera más sencilla de conectar ambas placas, se requerirá para ello de un cable USB con un conector tipo 'A' en un extremo, que se conectará a cualquiera de los puertos USB de la Raspberry Pi, y un conector tipo 'B' en el otro extremo que irá conectado al puerto USB del Arduino.

La longitud máxima que deberá tener este cable es de 3 metros.



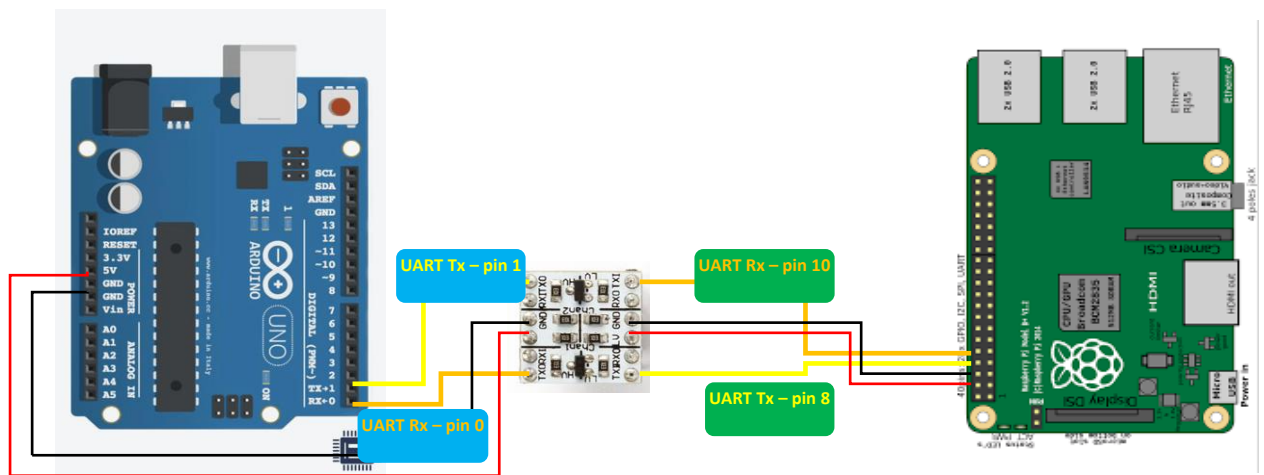
## Conexión via Terminales Tx/Rx

Lo primero que hay que mencionar en este caso, es que las placas manejan niveles lógicos distintos, el Arduino trabaja con voltajes de 0V y 5V (0 y 1 lógicos respectivamente) mientras que la Raspberry Pi opera con voltajes de 0V y 3.3V (0 y 1 lógicos respectivamente), por tanto no podremos conectar los puertos de ambas placas directamente.



Los pines UART en la Raspberry Pi son los pines 8 (Tx) y 10 (Rx) y en el Arduino son el pin 1 (Tx) y el pin 0 (Rx). La conexión entre ambos pines la realizamos a través de un componente electrónico denominado en inglés como Voltage Level Shifter, cuya función será la de convertir las señales de 5V a 3.3V y viceversa. En el caso específico de este montaje se utilizó un convertidor modelo WPI410 que posee dos entradas unidireccionales de 5 a 3.3 voltios, RXI a RXO, y dos terminales bidireccionales, TXI y TXO, de 5 a 3.3 voltios y viceversa. La alimentación del chip se realiza mediante dos conexiones de Vcc, una a 5V, terminal HV y la otra a 3.3V, terminal LV. La tierra de ambas placas van conectadas al terminal GND. El esquema de conexión es el que se muestra en la siguiente figura.





La conexión mediante estos terminales puede alcanzar los 15 metros.

## **CONFIGURACIÓN DE SOFTWARE**

### **CONFIGURACIÓN DE SOFTWARE DEL ARDUINO UNO**

En el caso del Arduino este ya viene preparado para trabajar con el puerto UART y no requiere que se realice ninguna configuración. En cuanto al software para las aplicaciones, se desarrollara en C++ y se utilizara la librería Serial para manejar las comunicaciones con los pines Tx y Rx. Esta librería se instala automáticamente cuando se instala el IDE de Arduino, por tanto tampoco en este aspecto se requiere realizar ninguna operación, únicamente tener instalado el Arduino IDE en la máquina de desarrollo, que puede ser la misma Raspberry Pi.

### **CONFIGURACIÓN DE SOFTWARE DE LA RASPBERRY PI**

Como primer paso debemos habilitar la comunicación Serial en los pines Tx y Rx de la Raspberry Pi (pines 8 y 10). Para ello ingresaremos a la Terminal de la Raspberry, bien sea directamente desde la placa o a través de Putty y ejecutaremos el comando.

#### **1) sudo raspi-config**

Se nos presentará la pantalla de configuración y procedemos a habilitar el Puerto Serial:

#### **2) Raspberry Pi Software Configuration Tool**

Seleccionaremos la opción 1: System options

Seleccionamos la opción 3: Interface Options

Seleccionamos la opción I6: Serial Port

Pregunta: Would you like a login Shell to be accesible over serial?:

<No>

Pregunta: Would you like the serial port hardware to be enable?:

<Yes>

Nos solicitará confirmación:

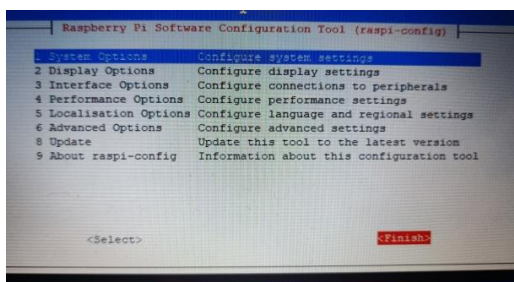
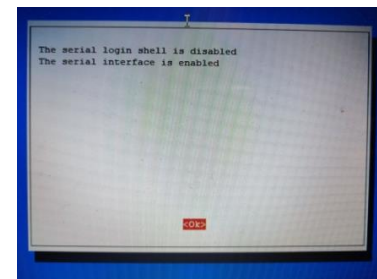
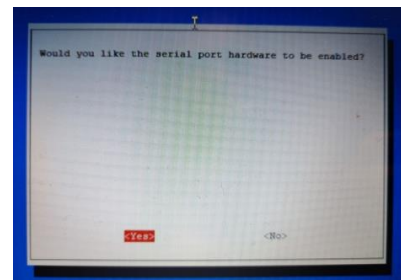
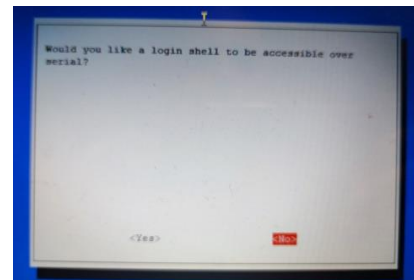
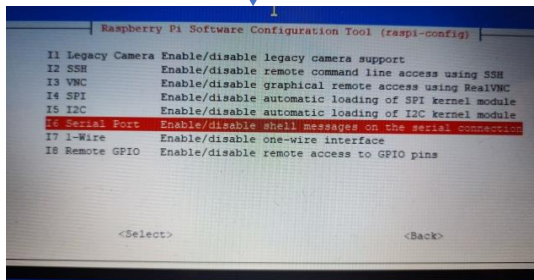
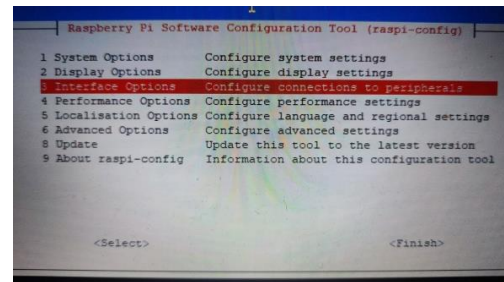
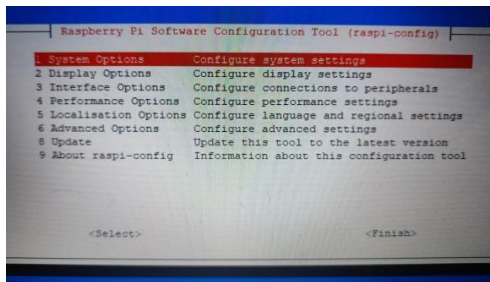
The Serial login shell is disabled

The serial interface in enabled

Y le indicaremos <Ok> y reinicializamos la Raspberry Pi para habilitar los cambios,

#### **3) Sudo reboot**

**sudo raspi-config**



**sudo reboot**

Una vez habilitado el puerto serial, debemos proceder con la reasignación del puerto serial al GPIO de la Raspberry Pi 3. Esta placa viene con dos puertos UART denominados:

## PL011 UART

### mini UART

Por defecto en la Raspberry Pi 3 el UART PL011 está asignado al puerto ttyAMA0 y el mini UART viene asignado al puerto ttyS0. A su vez, para mantener la compatibilidad con otras versiones, a estos puertos se les asignó el alias serial0 al puerto ttyAMA0 y el alias serial1 al ttyS0.

El puerto PL011 UART es un puerto basado en ARM y tiene mejor rendimiento que el mini UART. En la Raspberry Pi 3, el puerto mini UART es el que utiliza Linux para la consola, mientras que al puerto PL011 esta conectado el Bluetooth (en otras versiones de Raspberry, el Linux utiliza el puerto PL011 para la consola).

El puerto mini UART utiliza la frecuencia del GPU como referencia base para calcular su “baud rate”. Al variar la carga de trabajo, varía la frecuencia del GPU, lo que hace que varíe la frecuencia a la que opera el puerto mini UART, que a su vez hace que varíe la frecuencia de transmisión y recepción (baud rate) del puerto serial. Esto trae como consecuencia que el puerto sea inestable, lo que se traduce en pérdida de información y corrupción de la data. Para corregir esto en el proyecto, ya que el mismo basa todas las comunicaciones entre la Raspberry Pi y el Arduino en los puertos seriales Tx y Rx de ambas placas, se intercambiaron estos puertos y se asignó el puerto ttyAMA0 a los GPIO 14 y 15 (pines 8 y 10 o Tx y Rx) de la Raspberry y se reasignó el Bluetooth al puerto mini UART. El procedimiento para realizar este intercambio de puertos es el siguiente:

Abrir con el editor el archivo config.txt:

```
sudo nano /boot/config.txt
```

Agregar al final de archivo la instrucción:

```
dtoverlay = pi3-miniuart-bt
```

Guardar los cambios y Reiniciar el sistema

```
Sudo reboot
```

Para verificar que los cambios fueron exitosos, ejecutar la instrucción:

```
ls -l /dev
```

y deberán aparecer los puertos reasignados:

```
serial0 ----> ttyAMA0
```

```
serial1 ----> ttyS0
```

Como último paso es necesario instalar el software para el desarrollo de aplicaciones sobre la Raspberry Pi. Para el desarrollo de los módulos del proyecto que se ejecutaran en la Raspberry se utilizara Python 3.7 y la librería PySerial, que es la más utilizada y documentada y reúne todas las funcionalidades necesarias. El intérprete de Python se instala al instalar el Sistema Operativo Raspbian y en cuanto a la única librería necesaria, se procederá a ejecutar en la Raspberry Pi el comando:

## **pip3 install PySerial**

Con esto quedan configuradas las dos placas y ya podemos empezar con el desarrollo del sistema.

## **PRUEBAS INTEGRALES DE LA PLATAFORMA DE COMUNICACIONES.**

Antes de empezar con el desarrollo del sistema de IoT, debemos asegurarnos que hemos configurado correctamente el hardware y el software en ambas placas, tanto en la Raspberry Pi como en el Arduino Uno. Adicionalmente debemos asegurarnos que las conexiones entre ambas placas estén correctas y que las señales que llegan a los pines Tx y Rx de cada una de ellas a través del “Voltage Shifter” lo hacen con el voltaje adecuado.

La estrategia que seguiremos para ello será la siguiente:

- Realizaremos el envío de paquetes desde la Raspberry Pi al Arduino y verificaremos que llegue correctamente.
- Realizaremos el envío de paquetes desde el Arduino hacia la Raspberry Pi y verificaremos que llegue correctamente.
- Comprobaremos la comunicación Bidireccional

## **Prueba de Comunicación Serial desde el Arduino hacia la Raspberry Pi**

<b>Código en el Arduino</b>	<b>Código en la Raspberri Pi</b>
<pre>// PRUEBA DE COMUNICACION DESDE UNO HACIA PI // SE ENVIA UN CARACTER POR EL PUERTO SERIAL DEL ARDUINO Y SE VERIFICA SI // SI SE RECIBE EN LA RASPBERRY  void setup(){    Serial.begin(9600);   delay(3000);  }  void loop(){    Serial.println("Z"); // Enviamos un caracter por el puerto Serial</pre>	<pre># PRUEBA DE COMUNICACION DESDE PI HACIA UNO # SE ENVIA UN CARACTER POR EL PUERTO SERIAL DE LA RASPBERRY Y SE VERIFICA SI # SI SE RECIBE EN EL ARDUINO  import serial import time  if __name__ == '__main__':    puerto_serie = serial.Serial('/dev/ttyAMA0', 9600, timeout=1)   puerto_serie.reset_input_buffer()   time.sleep(3) # Esta espera es para permitir que el puerto serial del Arduino se estabilice</pre>

<pre> delay(3000); } </pre>	<pre> while True:      if (puerto_serie.inWaiting() &gt; 0):         mensaje = puerto_serie.readline()         x = mensaje.decode()         print('Raspberry Recibio desde el Arduino el mensaje: ', x)      print('Esperando...')     time.sleep(3) </pre>
-----------------------------	---

## Prueba de Comunicación Serial desde la Raspberry Pi hacia el Arduino

Código en el Arduino	Código en la Raspberri Pi
<pre> // PRUEBA DE COMUNICACION DESDE PI HACIA UNO // SE ENVIA UN CARACTER POR EL PUERTO SERIAL DE LA RASPBERRY Y SE VERIFICA SI // SI SE RECIBE EN EL ARDUINO  char mensaje;  void setup(){   Serial.begin(9600); }  void loop(){    if (Serial.available()) {      mensaje = Serial.read();     Serial.print("Arduino Recibio desde la Raspberry el mensaje: ");     Serial.println(mensaje);   }    delay(2000); } </pre>	<pre> # PRUEBA DE COMUNICACION DESDE PI HACIA UNO # SE ENVIA UN CARACTER POR EL PUERTO SERIAL DE LA RASPBERRY Y SE VERIFICA SI # SI SE RECIBE EN EL ARDUINO  import serial import time  if __name__ == '__main__':      puerto_serie = serial.Serial('/dev/ttyAMA0', 9600, timeout=1)     puerto_serie.reset_input_buffer()      time.sleep(3) # Esta espera es para permitir que el puerto serial del Arduino se estabilice      mensaje = b'A'      puerto_serie.write(mensaje)      print('Ya se envio a Arduino el mensaje: ', mensaje) </pre>

## Prueba de Comunicación Serial Bidireccional

Código en el Arduino	Código en la Raspberri Pi
<pre>// PRUEBA DE COMUNICACION DESDE PI HACIA UNO Y VICEVERSA // AMBAS PLACAS SE INTERCAMBIAN MENSAJES POR EL PUERTO SERIAL  String mensaje;  void setup(){  Serial.begin(9600); delay(3000);  }  void loop(){    if (Serial.available()) {      mensaje = Serial.readString();     delay(1000);     Serial.println("Hola PI me llamo UNO");   }    delay(2000); }</pre>	<pre># PRUEBA DE COMUNICACION DESDE PI HACIA UNO Y VICEVERSA # AMBAS PLACAS SE INTERCAMBIAN MENSAJES POR EL PUERTO SERIAL  import serial import time  if __name__ == '__main__':      puerto_serie = serial.Serial('/dev/ttyAMA0', 9600, timeout=1)     puerto_serie.reset_input_buffer()     time.sleep(3) # Tiempo de espera para estabilizacion del puerto serial del Arduino      while True:          saludo = 'Hola UNO soy PI'          puerto_serie.write(saludo.encode())          print('Ya me presente con Arduino y espero su respuesta')          print('Esperando...')          time.sleep(3)          if (puerto_serie.inWaiting() &gt; 0):              respuesta = puerto_serie.readline()              x = respuesta.decode()              print('UNO respondio: ', x)              time.sleep(3)</pre>

## **Etapas 2**

### **RECOLECCIÓN DE INFORMACIÓN Y ENVÍO A LA NUBE**

Un elemento importante para lograr el uso adecuado de los recursos y reducir el uso de fertilizantes, pesticidas y funguicidas sin poner en peligro la producción agrícola, es el empleo de modelos predictivos basados en Inteligencia Artificial. Estos modelos se alimentan de la información recolectada por los sensores y “predicen” las acciones necesarias para mantener la cosecha sana bajo esas condiciones, como por ejemplo, ventilar el área para reducir la temperatura o dispersar concentraciones de gases nocivos para las plantas o fumigar de manera preventiva dado un valor determinado de humedad relativa.

Una parte importante de la información que se requiere para alimentar el modelo predictivo son la Temperatura ambiente y la Humedad relativa. El proceso será:

1. El Coordinador solicitará al Nodo sensor la Temperatura ambiente y la Humedad relativa;
2. El Nodo sensor lee el mensaje y procede a leer los valores de Temperatura y Humedad del sensor;
3. El Nodo sensor empaqueta la información y la envía al Coordinador;
4. En Coordinador lee la información, la desempaqueta y la convierte en números para que pueda ser utilizada por el algoritmo de IA;
5. El Coordinador envía la data a la Nube.

Para el montaje de la Etapa 2 se utilizaron:

- 1 – Raspberry Pi como Nodo Coordinador
- 1 – Arduino Uno como Nodo Sensor
- 1 – Level Converter 3.3V/5V
- 1 – Sensor DHT11 para medir la Temperatura y la Humedad

En cuanto al software se utilizaron las siguientes librerías:

En el Nodo Coordinador (Raspberry Pi):

```
cayenne.client  
serial  
time
```

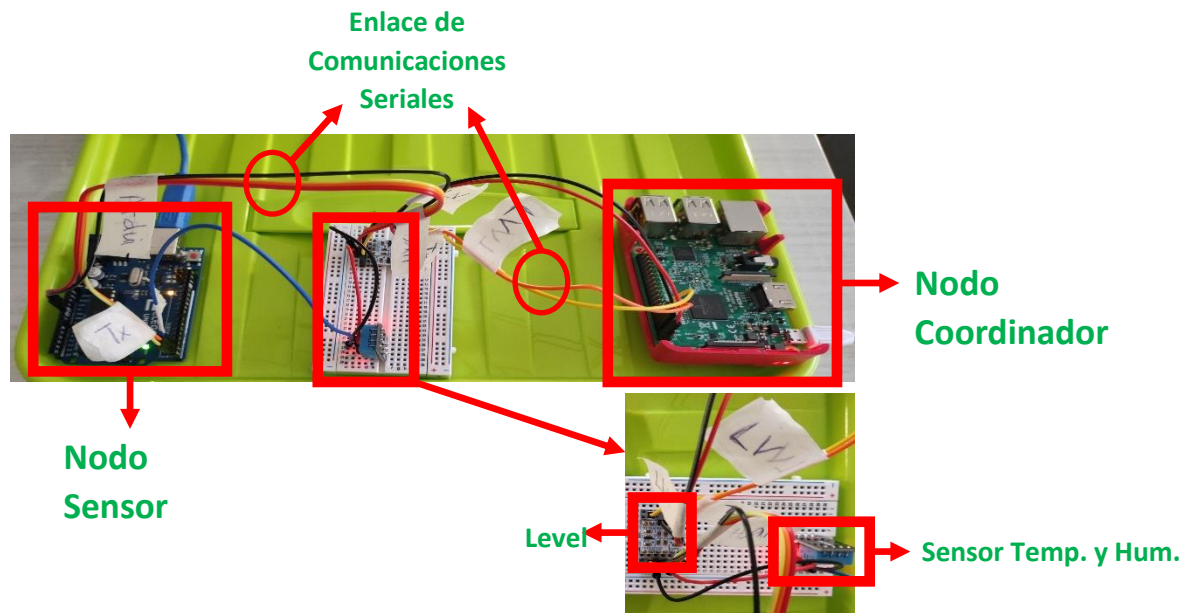
En el Nodo Sensor:

```
dht_nonblocking
```



El código que se ejecuta en ambos nodos se puede ver en el Anexo 1

El montaje electrónico quedo de la siguiente forma:



### Proceso de obtención de Datos (Sistema en operación)

**1 - El Coordinador solicitará al Nodo Sensor la Temperatura Ambiente y la Humedad Relativa**

L → Leer

a → ambiente (T y H)

**3 - El Nodo Sensor empaqueta la información y la envía al Coordinador**

T indica que el siguiente dato es la temperatura

; separador de datos

H indica que el siguiente dato corresponde a la

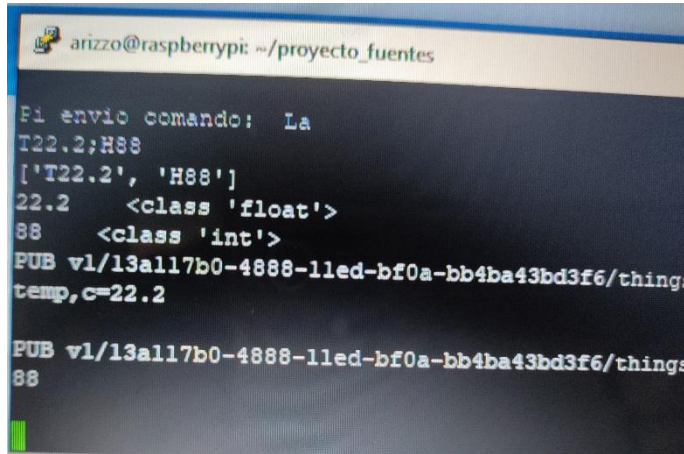
**4 - El Coordinador lee la información, la desempaqueta y la convierte en números para que pueda ser utilizada por el algoritmo de IA**

```
arizzo@raspberrypi: ~/proyecto_fuentes
Pi envio comando: La
T22.2;H88
['T22.2', 'H88']
22.2 <class 'float'>
88 <class 'int'>
PUB v1/13a117b0-4888-11ed-bf0a-b8
temp,c=22.2
PUB v1/13a117b0-4888-11ed-bf0a-b8
88
```

**5 - El Coordinador envía la data a la Nube**

## Ejecución en Nodo Coordinador (Raspberry Pi) y valores subidos a la Nube Cayenne

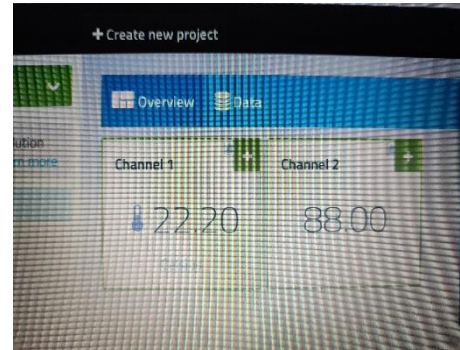
### Nodo Coordinador (Raspberry Pi)



```
arizzo@raspberrypi: ~/proyecto_fuentes
Pi envio comando: La
T22.2;H88
['T22.2', 'H88']
22.2    <class 'float'>
88      <class 'int'>
PUB v1/13a117b0-4888-11ed-bf0a-bb4ba43bd3f6/things
temp,c=22.2

PUB v1/13a117b0-4888-11ed-bf0a-bb4ba43bd3f6/things
88
```

### Cayenne



Las pruebas iniciales del sistema no resultaron exitosas, cuando se ejecutaba de manera aislada la librería `dht_nonblocking` para leer los valores de Temperatura y Humedad, se obtenían correctamente los valores, pero cuando la incorporábamos dentro del código del Nodo sensor (Arduino UNO), la misma no ejecutaba correctamente, devolviendo siempre un “false”, que indicaba que no había podido realizarse la lectura del sensor. Durante varios días se hicieron innumerables intentos infructuosos para tratar de incorporar la librería hasta que leímos en internet que dicha librería se debía ejecutar en el exacto ciclo de tiempo en la que fue programada y que cualquier delay, por pequeño que fuese, resultaba en una medida errónea del sensor. Entonces se procedió a buscar y probar otras librerías, resultando la adecuada para nuestro caso la librería `DHT.h` (DHT sensor library versión 1.4.4 del 27/06/2022).

En el video que se adjunta se puede ver el sistema trabajando.

### **Etapas 3**

En esta etapa se debe programar al nodo Actuador para recibir las instrucciones del Nodo Coordinador y enviar las órdenes correspondientes a los actuadores para que estos operen. Los principales actuadores de este subsistema son las bombas y electroválvulas para realizar el riego, fumigación y abono y los ventiladores para mover la masa de aire dentro del invernadero y controlar así la calidad del aire y la temperatura, así como la prevención de enfermedades.

Esta etapa no se pudo ejecutar como consecuencia de los problemas que se presentaron en las dos etapas anteriores y que nos consumieron todo el tiempo del proyecto, sin embargo es la mas fácil de implementar.

### **Posibles desarrollos futuros**

Establecer comunicaciones MQTT sobre LoRa entre Arduino y Raspberri para comunicaciones a distancias mayores para extensiones grandes de terreno. Para separar el recolector de data de los actuadores que deberán estarse desplazando

### **WEBgrafia**

NUEVAS TECNOLOGÍAS Y AGRICULTURA 4.0:

IMPACTO EN LOS RECURSOS HUMANOS DE LA INDUSTRIA AGRÍCOLA EN CENTROAMÉRICA

<https://repositorio.comillas.edu/xmlui/bitstream/handle/11531/46846/TFG%20-%20201916350.pdf?sequence=3&isAllowed=y>

Propuesta de una Arquitectura para Agricultura de Precisión Soportada en IoT

<https://pdfs.semanticscholar.org/4c6f/37715e4314c1089a009cb5899cbac9a67ce0.pdf>

SGreenH-IoT: Plataforma IoT para Agricultura de Precisión

<https://www.iiisci.org/journal/pdv/risci/pdfs/ca544si17.pdf>

## ANEXO 1

### **Código Fuente Nodo Coordinador (Raspberry Pi)**

```
# Python

import serial
import time
import Cayenne_lib_ar as cayenne

# ORDENES:
leer = 'L'
escribir = 'E'
regar = 'R'

# SENSORES:
s_ambiente = 'a'
s_suelo = 's'

# ACCIONES:
a_iniciar = 'i'
a_detener = 'd'

# FRECUENCIA DE LAS LECTURAS
# SE SIMULARA UN CICLO DE 1 DIA EN APROXIMADAMENTE 15 MINUTOS

f_ambiente = 10 # Equivale a 15 minutos)
f_suelo = 20    # Equivale a 30 minutos)

t_actual = 0
t_lectura_TH = 0

def main():

    # setup()
    # Configura Parametros del Puerto Serial y crea la instancia:
    puerto_serie = serial.Serial('/dev/ttyAMA0', 9600, timeout=1)
    puerto_serie.reset_input_buffer()
    time.sleep(3) # Tiempo de espera para estabilizacion del puerto serial del Arduino

    # Configura cuenta Cayenne para simulacion IA y crea la instancia:
    usuario= '13a117b0-4888-xxxx-bf0a-bb4ba43bd3f6'
    clave= 'd60a19917xxxxab5649fb1d82f61d7f576bf8ae8'
    id= '25a44c70-4888-11ed-xxxx-35fab7fd0ac8'

    obj_cayen= cayenne.inicia_conexion (usuario, clave, id)

# Inicializa otras c\variables
```

```

t_lectura_TH= time.time()

# loop()

while True:

# Solicita al Arduino los Valores de Temperatura y Humedad Ambiente cada f_ambiente segundos
# (que equivalen a 15 minutos):

    t_actual = time.time()
    if (t_actual - t_lectura_TH >= f_ambiente):
        comando = leer + s_ambiente
        time.sleep(0.1)
        puerto_serie.write (comando.encode('utf-8'))
        print ('Pi envio comando: ', comando)
        t_lectura_TH = time.time()

# Recibe Respuesta Arduino
time.sleep(1)
if (puerto_serie.inWaiting() > 0):
    time.sleep(0.1)
    mensaje = ((puerto_serie.readline()).decode()).rstrip('\r\n')

    print(mensaje)

    data = mensaje.split(';')
    print (data)
    for dato in data:
        if (dato[0] == 'T'):
            temperatura = float(dato.lstrip('T'))
        elif (dato[0] == 'H'):
            humedad = int(dato.lstrip('H'))
        else:
            print('Lecturas de Temperatura y Humedad con errores')

# Envia valores de Temperatura y Humedad Ambiente a Cayenne
# Para alimentar modelo IA

    print (temperatura, ' ', type(temperatura))
    print (humedad, ' ', type(humedad))
    obj_cayen.loop()
    cayenne.envia_datos(obj_cayen, temperatura, humedad)

    puerto_serie.close()

if __name__ == '__main__':
    main()

```

## **# Modulo Cayenne\_lib\_ar**

```
import cayenne.client
```

```
def inicia_conexion(usuario, clave, id):
```

```
# Crea el objeto cliente
```

```
    obj_conex = cayenne.client.CayenneMQTTClient()
```

```
    obj_conex.begin(usuario, clave, id)
```

```
    return obj_conex
```

```
def envia_datos(obj_conex, temp, humedad):
```

```
    obj_conex.virtualWrite(1,temp,"temp","c")
```

```
    obj_conex.virtualWrite(2,humedad)
```

## Código Fuente Nodo Sensor (Arduino UNO)

```
#include <DHT.h>

#define TipoSensor DHT11
int pinSensor = 2;

DHT SensorHT (pinSensor, TipoSensor);
float humedad;
float temperatura;
char comando;
char parametro;

void setup() {
  Serial.begin(9600);
  SensorHT.begin();
  delay(2000);
}

void loop() {

  // Lee Ordenes del coordinador
  if (Serial.available()) {
    delay(100);
    comando = Serial.read();
    delay(100);
    parametro = Serial.read();

    if (comando == 'L'){
//    Arduino Recibio el comando: LEER
      comando = "";
      if (parametro == 'a'){
//      Sensor de Temperatura y Humedad Ambiente
        parametro = "";

        // Lee T y H del sensor DHT
        humedad = SensorHT.readHumidity();
        temperatura = SensorHT.readTemperature();

        // Envia T y H al coordinador
        Serial.print("T");
        Serial.print(temperatura,1);
        Serial.print(";");
        Serial.print("H");
        Serial.println(humedad,0);

      }
    }
    delay(1000);
  }
}
```