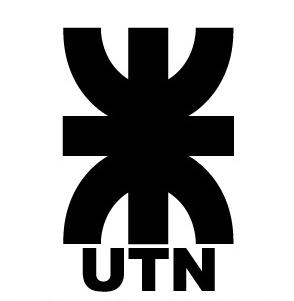
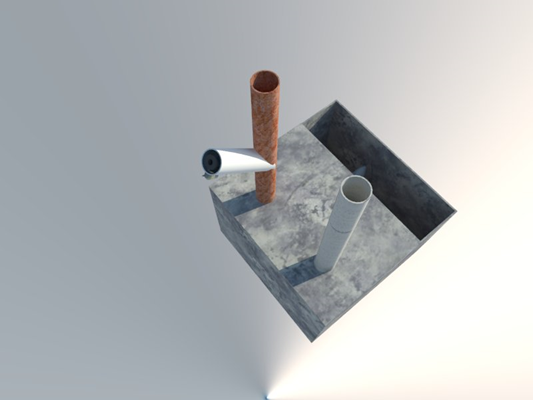
****

**Facultad Regional La Plata**

**PRACTICA SUPERVISADA 2017**

**Desarrollo de Módulo Automatizado para la producción de pollos a nivel familiar**

**SEGUNDO INFORME AVANCE**

****

**Alumno:** Tojo Adema, Hernan.

**Legajo:** 05-22601-5.

**Mail:** tojohernan@gmail.com

**Año:** 2018

**PERIODO AVANCE** : AGOSTO 2017 - MARZO 2018

ACTIVIDADES

En el periodo de trabajo se realizaron las siguientes actividades:

1. Compra de insumos
2. Realización de prototipado
   1. Instalación de y configuración de ESP en Arduino MEGA
   2. Comparativa Arduino vs Raspberry
   3. Configuracion del SO Raspberrian y placa Wifi
   4. Conexiones de Sensores
   5. Desarrollo y pruebas del software
3. Conclusión

INDICE

[**2) a - Instalación de y configuración de ESP en Arduino MEGA**](#_22a0u81gzk24) **3**

[ESP-01 la placa microcontroladora](#_igbfenisaenx) 3

[**2) b - Comparativa Arduino Vs Raspberry**](#_oc6nu77iiky) **4**

[**2) c- Configuracion del SO Raspberrian y placa Wifi**](#_a5uht8n87jqr) **5**

[**2) d - Conexiones de Sensores**](#_o8e5n3gaejw) **6**

[Conexion de sensor de temperatura](#_79lcctqysiq7) 6

[Conexión de sensor de distancia](#_vmotgrvsu21n) 7

[**2) e - Desarrollo y pruebas del software**](#_klstu8bcy36j) **8**

[Problemas de conexión a internet](#_aondf1qg17en) 9

[Control\_temperatura.py](#_6d0bj1tnd6yj) 9

[Control\_comedero.py](#_o7jptox82i9) 11

[Rpi\_a\_firebase.py](#_buh826vz9dw7) 12

[**6 - Conclusión**](#_ix1etrvwkpk8) **14**

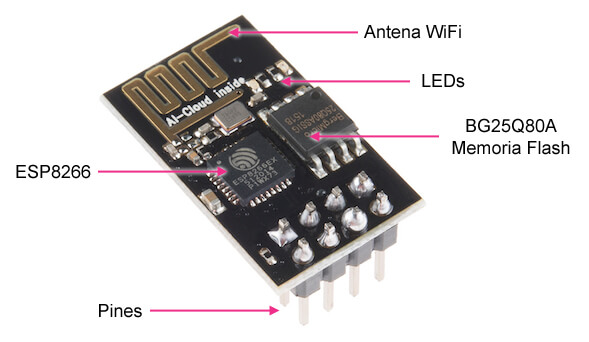
RESUMEN AVANCE

En este informe se detalla la construcción del prototipo tanto el Hardware como el Software y los cambios que fueron requeridos a lo largo del proyecto. También se realizará una conclusión donde se analiza la factibilidad en la implementación de un sistema de estas características.

# 

# 2) a - Instalación de y configuración de ESP en Arduino MEGA

## **ESP-01 la placa microcontroladora**



* ESP8266 es el microcontrolador del módulo ESP-01.
* Pines donde conectaremos la alimentación, sensores y transmisión de programa.
* BG25Q80A es la memoria flash donde residen los programas o sketchs. El ESP8266 no dispone de este tipo de memoria y por eso es un chip a parte.
* LEDs que nos informan de si está encendido o no y de la transmisión de datos (Tx y Rx).
* La antena WiFi para conectarse a una red/Internet.

La configuración del ESP8286 se realizó de forma exitosa en una primera instancia, se pudo realizar pruebas de conexión a través de una comunicación por el Puerto Serie del arduino con comandos AT.

Cuando se quiso incorporar diferentes funcionalidades que requerían el proyecto, tanto la lectura de datos a través de los sensores y el control de diferentes actuadores se pudieron observar algunas limitaciones a la hora de desarrollar el software en el lenguaje C.

No descartamos que sea posible hacerlo, sino que investigando un poco más llegamos a la conclusión que debíamos cambiar el Hardware por una placa que nos de la posibilidad de ser más estable y tambien escalable ya que el cliente proponia poder añadir otras funcionalidades como una cámara web y otros sensores adicionales.

# 2) b - Comparativa Arduino Vs Raspberry

Considero oportuno aclarar el porqué del cambio de hardware en el proyecto realizando una comparativa de las 2 placas en cuestión:

**Raspberry Pi** es una computadora completamente funcional, un dispositivo de sistema en chip (SoC), que se ejecuta en un sistema operativo Linux especialmente diseñado para ello, llamado Rasbian. Rasbian es el sistema operativo oficial para Raspberry Pi, donde otros sistemas operativos de terceros como Firefox OS, Android, RISC OS, Ubuntu Mate, etc. se pueden instalar en Pi, incluso la versión de Windows 10 también está disponible para Pi. Al igual que una computadora, tiene memoria, procesador, puertos USB, salida de audio, controlador gráfico para salida HDMI y, como se ejecuta en Linux, la mayoría de las aplicaciones de software de Linux se pueden instalar en él. Tiene varios modelos y revisiones como Raspberry Pi, Raspberry Pi 2, Raspberry Pi Modelo B + etc.

**Arduino** es un microcontrolador, que no es tan poderoso como Raspberry Pi, y puede considerarse como un componente del sistema informático, pero es un gran hardware para proyectos de electrónica. No necesita ninguna aplicación de SO y software para funcionar, solo necesitamos escribir unas pocas líneas de código para que funcione.

Estas cualidades, y el poco costo adicional que se incorpora al proyecto hicieron que se tome la desicion de cambiar el Hardware. Raspberry nos solucionaba el problema de la conectividad rápidamente y nos permitia desarrollar el código en un lenguaje de más alto nivel como PYTON que fue el elegido, aunque podríamos haber elegido cualquier otro.

# 2) c- Configuracion del SO Raspberrian y placa Wifi

# 

Se cargo el SO Raspberrian en una micro sd de 16 GB y se inicializo la raspberry

Configuracion de la micro sd para que sea de Lectura/ Escritura (https://raspberrypi.stackexchange.com/questions/68912/can-start-pi-but-get-errors-due-to-read-only-filesystem )

*sudo su [type password]*

*mount -o remount, rw /*

Configuracion de la placa wifi **EDIMAX USB** (https://www.andreagrandi.it/2014/09/02/how-to-configure-edimax-ew-7811un-wifi-dongle-on-raspbian/ ):

Edit /etc/network/interfaces and insert these configuration values:

*auto lo*

*iface lo inet loopback*

*iface eth0 inet dhcp*

*allow-hotplug wlan0*

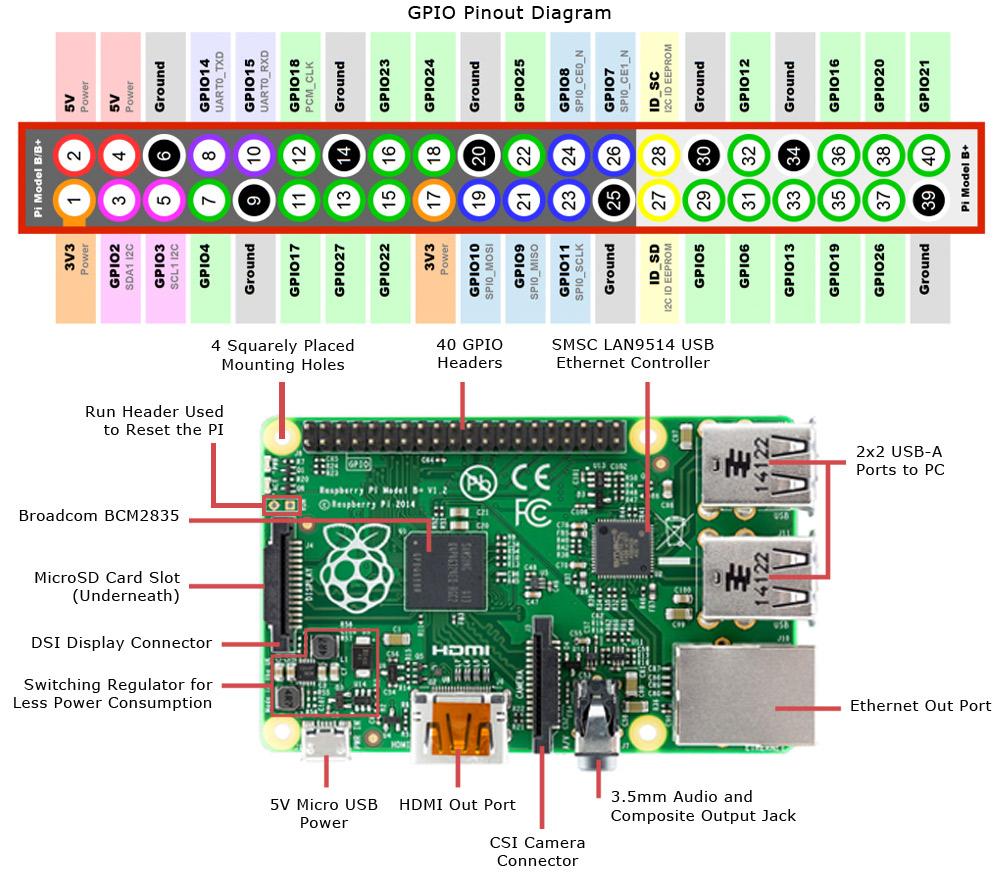
*auto wlan0*

*iface wlan0 inet dhcp*

*wpa-ssid YOURESSID*

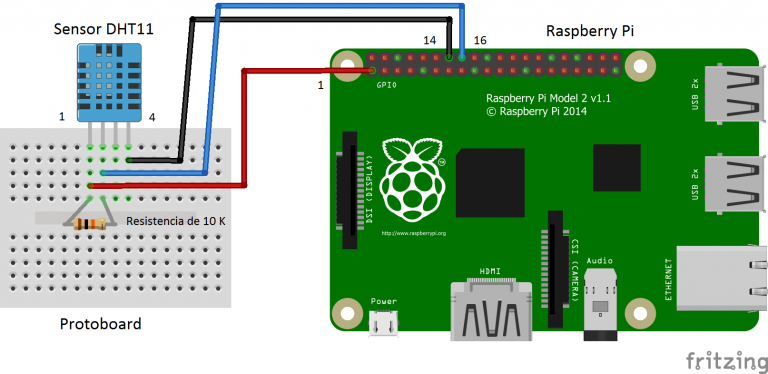
*wpa-psk YOURWPAPASSWORD*

Diagrama de conexiones disponibles en Raspberry



# 2) d - Conexiones de Sensores

### Conexion de sensor de temperatura

1. sudo apt-get update
2. sudo apt-get install git-core // instalar GIT
3. mkdir librerias // hacemos una carpeta donde estaran las librerias
4. git clone <https://github.com/adafruit/Adafruit_Python_DHT.git> // clonamos el proyecto Ardafruit que nos permite otener lecturas del sensor
5. sudo apt-get install build-essential python-dev // Instalamos y build essentials Python en el sistema
6. cd Adafruit\_Python\_DHT // nos posicionamos en la carpeta
7. sudo python setup.py install // instalamos la libreria Con esto la librería debería estar instalada y disponible para ser llamada desde cualquier programa en Python.
8. Realizamos el siguiente conexionado:
9. 

### 

### Conexión de sensor de distancia

## 

El sensor de distancia nos servirá para medir la cantidad de alimento que existe en el comedero, ubicado en la parte superior del mismo.

# 2) e - Desarrollo y pruebas del software

Se realizó el desarrollo del software en el lenguaje PYTON con la Base de datos Firebase para lo que se requirió de las siguientes librerías:

* Python // compilador para el lenguaje de desarrollo
* RPi.GPIO // para el control de las salidas donde se conectan sensores y actuadores
* Time // control de tiempo
* Adafruit\_Python\_DHT // Librería para controlar el sensor de temperatura DHT11
* Firebase // Administración de acceso y peticiones a la BD
* Git // Manejo de versionado de archivos y desarrollo colaborativo
* Urllib2 // para comprobar conexion a internet

### Problemas de conexión a internet

El software tiene como objetivo poder adquirir datos de sensores y realizar distintas funciones donde se automatiza el control de temperatura del módulo, otorgando más calor o ventilando en tiempo real, pero también es importante poder controlar estas acciones de forma remota, para ello los datos deben estar disponibles en alguna base de datos que en este caso será Firebase, donde podemos tener problemas para la carga de datos por lo que se implementara un script que detecte si no hay conexión, guarde los datos localmente y los envía cuando se recupera esa conexión.

### Control\_temperatura.py

Este script realiza un control en tiempo real de la temperatura del módulo:

1. Realiza una medición de la temperatura actual
2. Compara la temperatura actual con la temperatura ideal seteada previamente
3. En el caso de una sobre temperatura enciende la ventilación ( Representada por un LED Rojo) y en caso de un déficit de temperatura enciende una fuente de calor ( representada por un LED Azul)

//////////////////////////////////////// INICIO CODIGO /////////////////////////////////

*import Adafruit\_DHT*

*import RPi.GPIO as GPIO*

*GPIO.setmode(GPIO.BCM)*

*GPIO.setup(17, GPIO.OUT) ## GPIO 17 como salida*

*GPIO.setup(27, GPIO.OUT) ## GPIO 27 como salida*

*pin\_ventilacion=17*

*pin\_luz=27*

*GPIO.output(pin\_ventilacion, GPIO.LOW)*

*GPIO.output(pin\_ventilacion, GPIO.LOW)*

*sensor = Adafruit\_DHT.DHT11*

*pin\_dht11 = 23*

*temperatura\_ideal = 20*

*ventilacion\_prendida=False*

*luz\_prendida=False*

*def estado(sensor,accion):*

*if sensor == "v":*

*if accion == 1:*

*GPIO.output(pin\_ventilacion, GPIO.HIGH)*

*print ("PIN VENTILADOR ON")*

*return()*

*else:*

*GPIO.output(pin\_ventilacion, GPIO.LOW)*

*print ("PIN VENTILADOR OFF")*

*return()*

*else:*

*if accion == 1:*

*GPIO.output(pin\_luz, GPIO.HIGH)*

*print ("PIN LUZ ON")*

*return()*

*else:*

*GPIO.output(pin\_luz, GPIO.LOW)*

*print ("PIN LUZ OFF")*

*return()*

*while True:*

*humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin\_dht11)*

*#print("Cual es la temperatura actual?")*

*#temperature = input()*

*print (humidity)*

*print (temperature)*

*if temperature == temperatura\_ideal:*

*estado("l", 0)*

*luz\_prendida=False*

*estado("v", 0)*

*ventilacion\_prendida=False*

*else:*

*if (temperature < temperatura\_ideal-2) and (luz\_prendida==False):*

*print("Aumentando la temperatura..")*

*estado("l",1)*

*luz\_prendida=True*

*else:*

*if (temperature > temperatura\_ideal+2) and (ventilacion\_prendida == False):*

*print("Bajando la temperatura..")*

*estado("v",1)*

*ventilacion\_prendida=True*

*////////////////////////////// FIN CODIGO ////////////////////////////*

### Control\_comedero.py

El script realiza un seguimiento en tiempo real del porcentaje de alimento que se encuentra en el comedero, dependiendo el tamaño del mismo seteado previamente podremos determinar el porcentaje:

1. Consulta al sensor de distancia ubicado en la parte superior del comedero
2. Por medio de una fórmula matemática podremos deducir el porcentaje de alimento que se encuentra en el comedero y lo imprime en pantalla.

*////////////////////////////// INICIO CODIGO ////////////////////////////*

*import RPi.GPIO as GPIO #Importamos la libreria GPIO*

*import time #Importamos time (time.sleep)*

*GPIO.setmode(GPIO.BCM) #Ponemos la placa en modo BCM*

*GPIO\_TRIGGER = 25 #Usamos el pin GPIO 25 como TRIGGER*

*GPIO\_ECHO = 7 #Usamos el pin GPIO 7 como ECHO*

*GPIO.setup(GPIO\_TRIGGER,GPIO.OUT) #Configuramos Trigger como salida*

*GPIO.setup(GPIO\_ECHO,GPIO.IN) #Configuramos Echo como entrada*

*GPIO.output(GPIO\_TRIGGER,False) #Ponemos el pin 25 como LOW*

*try:*

*while True: #Iniciamos un loop infinito*

*GPIO.output(GPIO\_TRIGGER,True) #Enviamos un pulso de ultrasonidos*

*time.sleep(0.00001) #Una pequea pausa*

*GPIO.output(GPIO\_TRIGGER,False) #Apagamos el pulso*

*start = time.time() #Guarda el tiempo actual mediante time.time()*

*while GPIO.input(GPIO\_ECHO)==0: #Mientras el sensor no reciba seal...*

*start = time.time() #Mantenemos el tiempo actual mediante time.time()*

*while GPIO.input(GPIO\_ECHO)==1: #Si el sensor recibe senal...*

*stop = time.time() #Guarda el tiempo actual mediante time.time() en otra variable*

*elapsed = stop-start #Obtenemos el tiempo transcurrido entre envio y recepcion*

*distance = (elapsed \* 34300)/2 #Distancia es igual a tiempo por velocidad partido por 2 D = (T x V)/2*

*print distance #Devolvemos la distancia (en centimetros) por pantalla*

*time.sleep(1) #Pequena pausa para no saturar el procesador de la Raspberry*

*porcentaje\_llenado = ( 1000 - (distance \* 100)) / 9 #calculamoe le porcentaje*

*print("PORCENTAJE:")*

*print porcentaje\_llenado*

*except KeyboardInterrupt:*

*print "quit"*

*GPIO.cleanup()*

*////////////////////////////// FIN CODIGO ////////////////////////////*

### Rpi\_a\_firebase.py

El siguiente script realiza una autenticación en la base de datos Firebase previamente configurada y posteriormente envía diferentes datos de relevancia para que puedan ser vistos por cualquier usuario de internet o usados para un próximo desarrollo de alguna aplicación de administración del módulo.

*////////////////////////////// INICIO CODIGO ////////////////////////////*

*import json*

*import time*

*import urllib2*

*from firebase import firebase*

*import Adafruit\_DHT # Para leer la temperatura del DHT11*

*#### Inicializo variables ####*

*firebase = firebase.FirebaseApplication('https://comedero-9f39f.firebaseio.com/', None)*

*sensor = Adafruit\_DHT.DHT11*

*pin\_dht11 = 23*

*data\_backup = [ ]*

*def internet\_connection():*

*try:*

*urllib2.urlopen('http://www.google.com', timeout=1)*

*return True*

*except urllib2.URLError as err:*

*return False*

*def cargar\_backup():*

*n=0*

*for data in data\_backup:*

*n=n+1*

*result = firebase.post('mediciones/', data)*

*print (result)*

*print ("Se cargaron ", n ," datos de backup...")*

*def control\_temperatura(): # Necesito hacer el control y la lectura por separados*

*datos\_sin\_enviar=False*

*global data\_backup*

*while True:*

*humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin\_dht11)*

*data\_new = {*

*"ts": time.time(),*

*"temp": temperature,*

*"hum": humidity*

*}*

*if internet\_connection() == True:*

*if datos\_sin\_enviar==True:*

*cargar\_backup()*

*datos\_sin\_enviar=False*

*data\_backup= []*

*with open('data\_backup.json', 'w') as f:*

*json.dump(data\_backup, f)*

*f.close()*

*print (data\_backup)*

*print('SE VACIA BACKUP')*

*if humidity is not None and temperature is not None:*

*result = firebase.post('mediciones/', data\_new)*

*print(result)*

*print(data\_new)*

*else:*

*print('Fallo en la lectura!')*

*else:*

*datos\_sin\_enviar=True*

*data\_backup.append(data\_new)*

*with open('data\_backup.json', 'w') as f:*

*json.dump(data\_backup, f)*

*f.close()*

*print('Se cargo nueva data en backup: ')*

*print (data\_backup)*

*time.sleep(10)*

*try:*

*control\_temperatura()*

*except KeyboardInterrupt: #Si el usuario pulsa CONTROL+C...*

*print "quit" #Avisamos del cierre al usuario*

*////////////////////////////// FIN CODIGO ////////////////////////////*

# 6 - Conclusión

El prototipo obtenido aunque no es funcional para posibles pruebas, demuestra que podría realizarse un prototipo funcional de muy bajo costo para la realización de pruebas en territorio y mejorarlo hasta llegar a un producto final que optimice el proceso de producción de pollos parrilleros a nivel familiar.

Se deberían realizar las siguientes pruebas:

1. Determinar si los sensores son fieles a las mediciones reales.
2. Determinar si los sensores soportan el clima hostil de lugares alejados durante un periodo de tiempo determinado.
3. Alcanza con un sistema de conectividad Wifi o deberían incorporarse otros medios de comunicación.
4. El consumo de energía del módulo es adecuado o excesivo.
5. Cual es el retorno de la inversión (ROI) de un módulo funcional.

En el desarrollo se encontró otro uso además del de asistir al micro productor agropecuario que está referido a la investigación y educación. Si en universidades, escuelas agropecuarias, instituciones especializadas se instalarán este tipo de módulos podrían realizarse tareas educativas y proyectos de investigación donde se realicen pruebas con diferentes especificaciones en cuanto a las temperaturas, horas de luz, periodos de alimentación , etc. que tengan como objetivo optimizar el proceso productivo y pueda aplicarse a grandes producciones.