ANEXO IV

Domótica

Tecnologías de Control para Domótica

KNX

KNX es un sistema que ha sido desarrollado para manejar de manera automática y programada diferentes estancias del hogar y otras edificaciones.

La domótica KNX es un sistema BUS, reciben este nombre por el tipo de cableado que se emplea en ellos. Estos cables permiten una constante intercomunicación de unos con otros.

Los componentes individuales de un sistema KNX incluyen sensores y actuadores. Los sensores, como es el caso de los termostatos, conmutadores o anemómetros, generan comandos en forma de telegramas. Los actuadores (p. ej., relés de conmutación para cortinas o luces) transforman estos telegramas en acciones. Una línea bus de dos hilos proporciona la conexión y, por ende, el tráfico de telegramas entre sensores y actuadores. Con ello, se acaba con la necesidad de conectar los componentes individuales del sistema a la red, lo que reduce considerablemente el cableado.

Entre los usos más comunes de la domótica KNX, se tiene el control de iluminación, automatización de persianas, diseños de calefacción eficientes, etc.

KNX está aprobado como estándar internacional (ISO/IEC 14543-3), estándar europeo (CENELEC EN 50090 y CEN EN 13321-1) así como estándar nacional en países como China (GB/T 20965). Ello asegura la continuidad de KNX en el futuro. Dispositivos KNX de diferentes fabricantes pueden ser combinados - la marca registrada KNX garantiza la interoperabilidad y el

«interworking». En resumen, KNX es un estándar abierto líder a nivel mundial para el control tanto de viviendas como de edificios.

Z-Wave

Z-Wave es un protocolo de comunicaciones inalámbricas, desarrollado especialmente para la domótica. Es una red de malla, que utiliza ondas de radio de baja energía en una frecuencia de 800-900 MHz. Está diseñado para asegurar una transmisión fiable y de baja latencia de pequeños paquetes de datos. Otra característica importante de este protocolo es la interoperabilidad; esto significa que los dispositivos certificados Z-Wave son siempre compatibles entre sí.

En una red Z-Wave, cada dispositivo representa un nodo, y cada dispositivo está conectado a todos los demás dispositivos a su alcance. Esta configuración tiene dos ventajas principales:

No es necesaria la conexión a través de una pasarela – en una red Wifi, cada dispositivo tiene que conectarse primero a un dispositivo común (normalmente el router) para obtener acceso a la red. En una red Z-Wave, un nuevo dispositivo se conecta directamente a otros dispositivos y obtiene acceso instantáneo a la red.

Alcance – Los dispositivos Z-Wave tienen un alcance de unos 30 metros (incluso más con versiones más recientes), pero el alcance de toda la red Z-Wave es mucho más amplio. Este protocolo incluye salto de información de nodo a nodo hasta cuatro veces. Esto permite la comunicación entre dos dispositivos, que de otro modo estarían fuera del alcance del otro.

Estas propiedades distinguen a Z-Wave de otros protocolos de comunicación más utilizados, como Wifi, y la hacen más apropiada para su uso con aplicaciones de control y sensores.

Zigbee

ZigBee corresponde al nombre de la especificación creada por el grupo de trabajo ZigBee Alliance para la definición de un protocolo de comunicaciones inalámbrico de alto nivel.

Este protocolo se basa en el estándar 802.15.4 para las capas bajas de comunicación y tiene como principales rasgos diferenciales las siguientes características: Bajo consumo y baja tasa de transferencia de datos, topología de red tipo mesh, tamaño reducido.

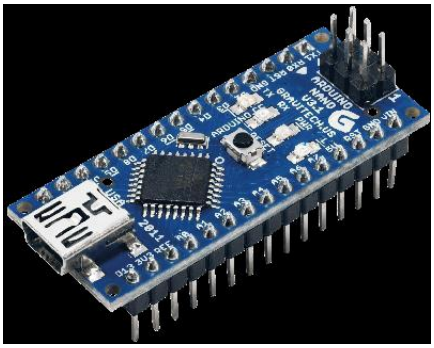
Gracias a estas características, ZigBee se ha posicionado como una de las mejores soluciones inalámbricas para equipos de control desplegados en campo como equipos de metering o sensórica, ya que pueden operar mediante baterías. Además, el protocolo ZigBee también gestiona de forma automática la red, creando y manteniendo las rutas de forma dinámica. Esto permite que la red corrija automáticamente situaciones de desconexión parcial de alguno de los nodos que la forman, evitando así tener que acceder físicamente a los equipos para poder solventar ese tipo de problemas.

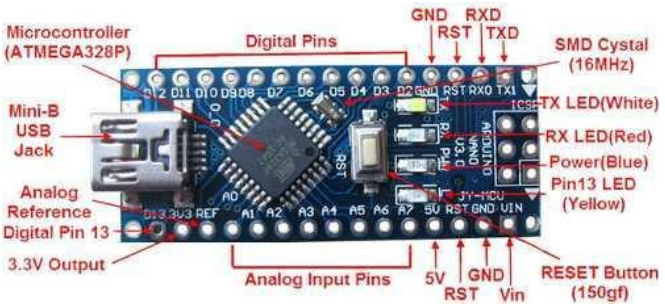
Otro de los campos en los que ZigBee es comúnmente usado es en aplicaciones domóticas, debido a su reducido tamaño y coste, lo que permite integrar módulos ZigBee en consumibles y equipos del hogar, como bombillas, electrodomésticos, alarmas y otros elementos comunes en una casa domótica.

Datos de Arduino, Modelos Arduino y sus características

Arduino Nano

Ilustración 1 - Placa Arduino Nano





La placa Arduino Nano es una placa de prueba pequeña y completa basada en ATmega328. Tiene funcionalidad similar al modelo Arduino Duemilanove, pero en un módulo DIP. Solo carece de jack de alimentación DC y funciona con un cable Mini-B USB en lugar de uno estándar.

Las características de entrada salida son que cada uno de los 14 pines digitales del Nano pueden ser usados como entrada o salida, usando las funciones pinMode(), digitalWrite(), y digitalRead(). Operan a 5 voltios. Cada pin puede proveer o recibir un máximo de 40mA y poseen una resistencia de pull-up (desconectada por defecto) de 20 a 50 kOhms.

Además, algunos pines poseen funciones especializadas:

Serial: 0 (RX) y 1 (TX). (RX) usado para recibir y (TX)usado para transmitir datos TTL vía serie. Estos pines están conectados a los pines correspondientes del chip USB-a-TTL de FTDI.

Interrupciones Externas: pines 2 y 3. Estos pines pueden ser configurados para activar una interrupción por paso a nivel bajo, por flanco de bajada o flanco de subida, o por un cambio de valor. Investigar la función attachInterrupt() para más detalles.

PWM: pines 3, 5, 6, 9, 10, y 11. Proveen de una salida PWM de 8-bits cuando se usa la función analogWrite(). SPI: pines 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estos pines soportan la comunicación SPI, la cual, a pesar de poseer el hardware, no está actualmente soportada en el lenguaje Arduino. LED: Pin 13. Existe un LED conectado al pin digital 13. Cuando el pin se encuentra en nivel alto, el LED está encendido, cuando el pin está a nivel bajo, el LED estará apagado.

El Nano posee 8 entradas analógicas, cada una de ellas provee de 10 bits de resolución (1024 valores diferentes). Por defecto miden entre 5 voltios y masa, sin embargo, es posible cambiar el rango superior usando la función analogReference(). También, algunos de estos pines poseen funciones especiales:

I2C: Pines 4 (SDA) y 5 (SCL). Soporta comunicación I2C (TWI) usando la librería Wire.

Hay algunos otros pines en la placa:

AREF. Tensión de referencia por las entradas analógicas. Se configura con la función analogReference().

Reset. Esta línea se utiliza a nivel bajo para resetear el microcontrolador. Normalmente se usa para añadir un botón de reset que mantiene a nivel alto el pin reset mientras no es pulsado.

Las características más destacadas son:

Microcontrolador ATmega328 con cargador de inicio pre programado.

Tensión de entrada (recomendada): +7 a + 12 V.

Tensión de entrada (límites): +6 a + 20 V.

14 pines GPIO (de los que 6 ofrecen salida PWM).

6 pines de entrada analógica.

Corriente DC por pin de E/S: 40 mA.

Memoria Flash de 32 KB (2 KB para cargador de inicio).

SRAM de 2 KB.

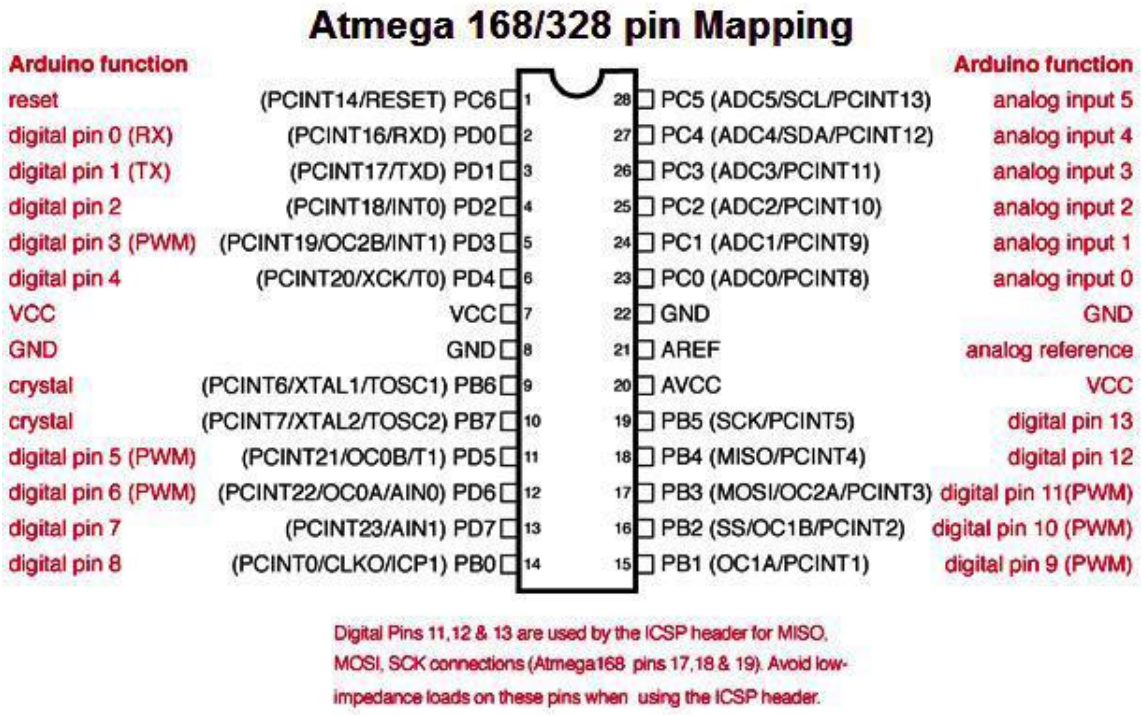
EEPROM de 1 KB.

Admite comunicación serie IC.

Frecuencia de reloj: 16 MHZ.

Dimensiones: 0,73″ x 1,7″.

Ilustración 2 - Característica de placa



NodeMCU

El NodeMCU es el módulo más característico de este tipo. Su precio ronda los 6€. A diferencia de los otros módulos, viene con todo lo necesario para empezar a trabajar de forma autónoma. Incluye un adaptador serie/USB y se alimenta a través del microusb. Está basado en el ESP-12 y la última versión oficial es la 2. Lo más interesante de este módulo es que se puede descargar un firmware que permite programar en lenguajes como LUA, Python, Basic o JavaScript. Sin duda alguna este módulo es la mejor opción si se pretende adentrarse en el mundo del ESP8266.

NodeMCU es la placa de desarrollo basada en el ESP8266 que facilita mucho el desarrollo de dispositivos conectados.

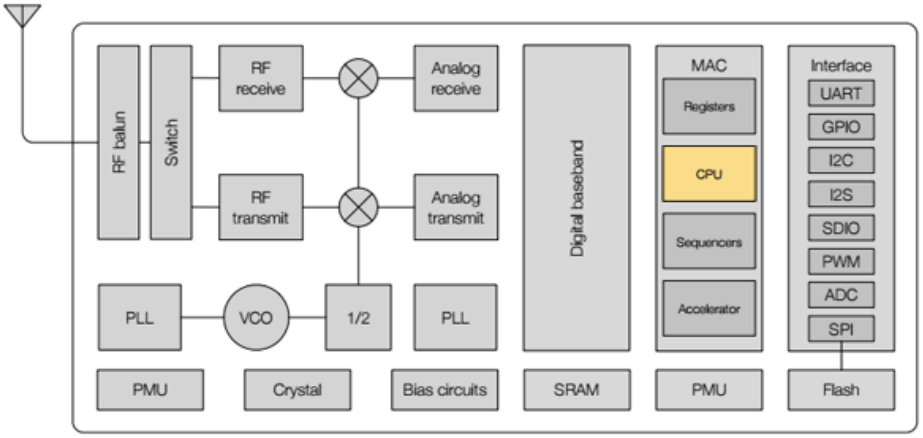
Ilustración 3 - Placa NodeMCU

Es una placa de desarrollo totalmente abierta, a nivel de software y de hardware. Al igual que ocurre con Arduino, en NodeMCU todo está dispuesto para facilitar la programación de un microcontrolador o MCU (del inglés Microcontroller Unit).

Son el primer paso hacia el Internet de las Cosas o el IoT. Pueden enviar datos, recibir datos e incluso controlar los pines de entrada y salida de forma remota e inalámbrica.

En términos estrictos el ESP8266 no es un microcontrolador. Si que lleva uno y se llama Tensilica L106 de 32-bit. La MCU se va a encargar de gestionar todas las entradas, salidas y cálculos necesarios para hacer funcionar el programa que se haya cargado.

Ilustración 4 - Datasheet



Funciona con 32-bit lo que viene a decir que puede realizar operaciones con números de ese tamaño (de 0 a 4.294.967.295 o de -2.147.483.648 a 2.147.483.647).

SoC ESP8266

Teniendo en claro qué es una MCU o microcontrolador, ahora se verá que es en realidad el ESP8266. El nombre técnico es ESP8266EX.

Ilustración 5 - SoC ESP8266

Se trata de un SoC o Sistema en Chip. Básicamente consiste en un chip que tiene todo integrado (o casi todo) para que pueda funcionar de forma autónoma como si fuera un ordenador. En el caso del ESP8266 lo único que no tiene es una memoria para almacenar los programas.

Esto supone un inconveniente ya que parte de los pines de entrada y salida, tendrán que ser utilizados para conectarse a una memoria Flash externa.

Las características principales son las siguientes:

Incorpora una MCU de 32-bit de bajo consumo (Tensilica L106)

Módulo WiFi de 2.4 GHz

RAM de unos 50 kB

1 entrada analógica de 10-bit (ADC)

17 pines de entrada y salida GPIO (de propósito general)

Kit o placa de desarrollo NodeMCU

Se encuentra ya al nivel superior donde todo es mucho más sencillo. Este tipo de kit incorpora componentes que ayudan a programar y conectar el módulo y la MCU a los circuitos.

Existen diferentes modelos de varias marcas con características y funcionalidades diferentes dependiendo del SoC o microcontrolador que utilizan.

Pero todos tienen el mismo objetivo, facilitar el prototipado y desarrollo de proyectos con microcontroladores.

NodeMCU es uno de ellos y sus características principales son:

Conversor Serie-USB para poder programar y alimentar a través del USB

Fácil acceso a los pines

Pines de alimentación para sensores y componentes

LEDs para indicar estado

Botón de reset

En este nivel, ya no es una preocupación el cómo cargar el programa o cómo conectar los pines. Ahora solo se necesita un ordenador, un cable USB y un entorno de desarrollo para programar la MCU o microcontrolador.

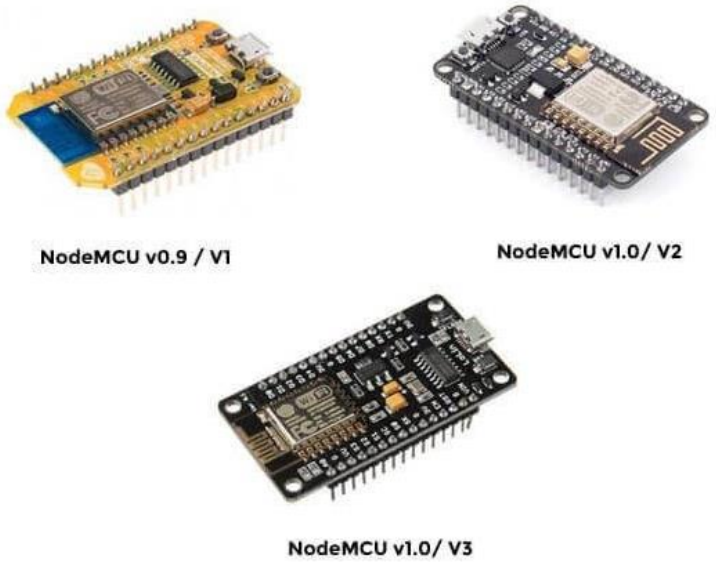
Versiones de NodeMCU

Hay diferentes versiones de NodeMCU. Todo esto es debido a que se trata de una placa de hardware abierto y cualquier fabricante puede crear su propia distribución.

Todos los NodeMCU se basan en los mismos módulos el ESP-12 y ESP-12E que a su vez se basan en el SoC ESP8266.

Partiendo de esta premisa, las diferencias que se pueden encontrar son básicamente el número de pines a los que se tiene acceso y el tamaño de cada placa.

Ilustración 6 - Placas Arduino Actuales



Actualmente hay varias versiones en el mercado, en concreto 3. Existe cierta confusión al respecto y es debido a que se han nombrado de diferentes maneras. Podemos encontrar el nombre dependiendo de la generación a la que pertenecen, la versión o el nombre común que se les ha dado.

Comparando las diferentes versiones de NodeMCU

Cuando se adquiere un NodeMCU es importante reconocer la versión a la que pertenece. Es fundamental a la hora de programar y a la hora de sacar todo el potencial.

Las placas de 1ª y 2ª generación son fáciles de distinguir debido a su tamaño. Ambas utilizan el módulo ESP-12 con 4 MB de Flash, pero la 2ª generación utiliza la más nueva y mejorada ESP-12E.

Ilustración 7 - Placa ESP-12E

En detalle cada una de las versiones de NodeMCU son:

1ª generación / v0.9 / V1

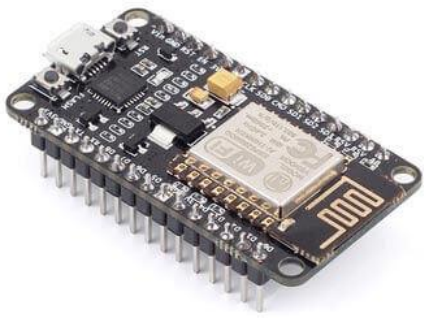
Ilustración 8 - Placa NodeMCU -V1

Quizás el mayor inconveniente que se encuentra con esta placa es su tamaño. Es muy incómodo cuando se conecta por ejemplo a una protoboard ya que ocupa todo el espacio y no deja hueco para conectar los pines.

El módulo que utiliza es el ESP-12 basado en el ESP8266 con una memoria Flash de 4 MB.

2ª generación / v1.0 / V2

Ilustración 9 - Placa NodeMCU -V2



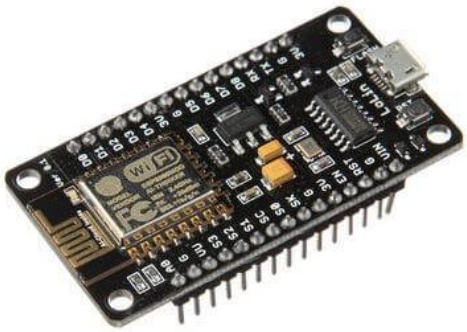
La gran diferencia, como ya se ha hecho, es el tamaño. Esta 2ª generación sí que encaja perfectamente en una protoboard y permite conectar cables a los pines de una forma sencilla.

Otra diferencia es que el chip fue actualizado del ESP-12 al ESP-12E y esto permite tener algunos pines extra.

Con esta nueva generación de NodeMCU se produjo un salto sustancial de calidad y facilidad en la programación y prototipado.

2ª generación / v1.0 / V3

Ilustración 10 - Placa NodeMCU -V3



Es una especificación oficial por parte de NodeMCU.

Se trata de una versión creada por Lolin que aporta ciertas mejoras como que el puerto USB parece ser más robusto y dos de los pines que no se utilizan en la versión V2 se han utilizado como salida de 5V directa del USB y un GND adicional.

Otra diferencia mínima es el tamaño ya que la V3 es un poco más ancha que la V2.

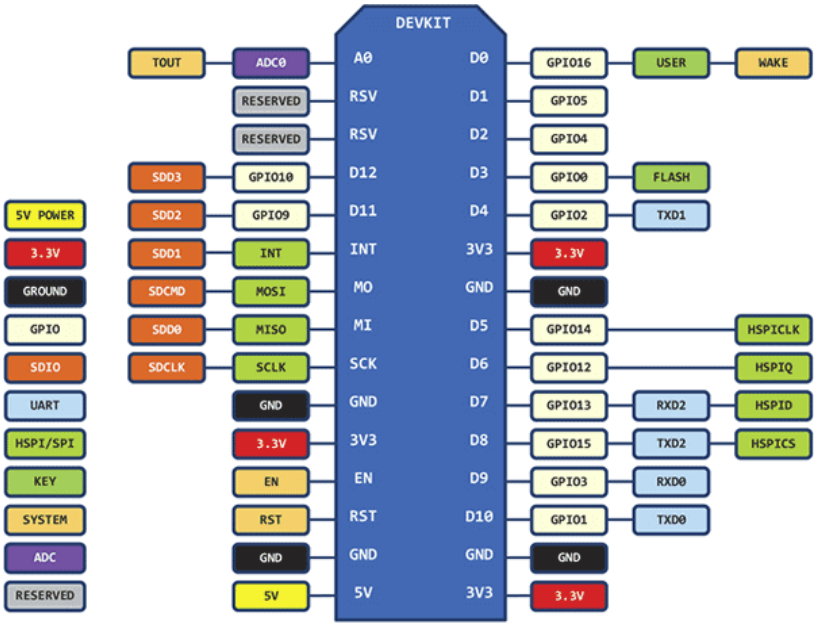
En el proyecto asociado a este anexo se utiliza la placa NodeMCU V2 ya que es la última versión oficial, y por lo cual se tomarán como validad las especificaciones de esta placa

Acceso a los pines de NodeMCU V2

Habiendo visto qué es el NodeMCU y las versiones que existen, se comenzara describiendo las características más importantes que aporta esta placa a nivel de hardware, los pines digitales, analógicos y de alimentación.

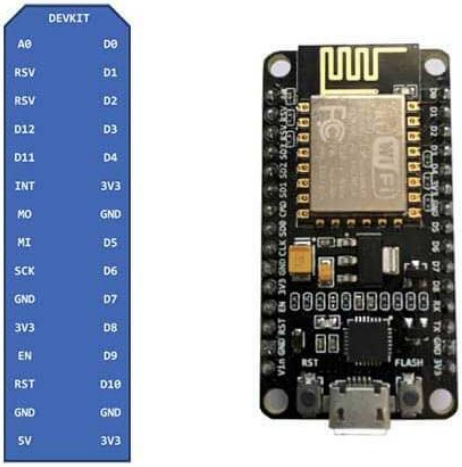
En la siguiente imagen se puede ver una visión general de todos los pines de NodeMCU V2.

Ilustración 11 - Pines Placa NodeMCU



Como norma, siempre que haga referencia a este diagrama de pines de NodeMCU, equivaldrá a que si se pondria la placa con el puerto USB hacia abajo como se muestra en la siguiente imagen.

Ilustración 12 - Ejemplo de ubicación

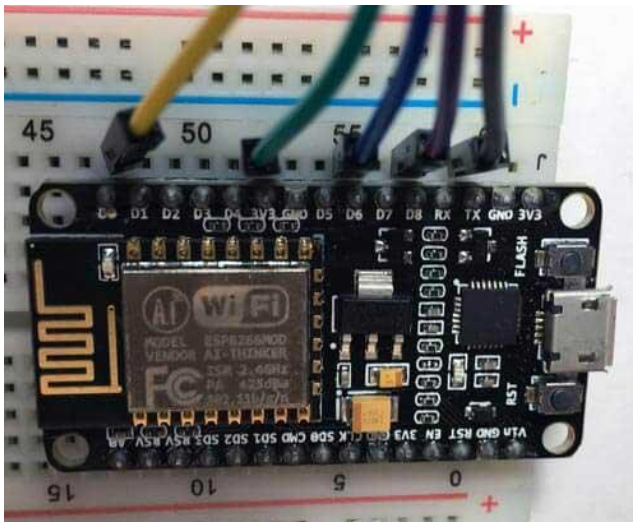


Así se sabrá en todo momento la correspondencia con los pines de la placa de desarrollo o prototipado.

Pines de NodeMCU

Gracias a la placa de desarrollo NodeMCU es posible acceder a los pines de una forma sencilla. Lo más práctico es pinchar la placa en una protoboard y desde aquí, sacar cables para conectarlos a los diferentes sensores y componentes, aunque siempre está la opción de conectarlos directamente al pin correspondiente de la misma placa.

Ilustración 13 - Conexión genérica de ESP 8266



Aunque se tiene acceso a más pines que los digitales, el pin analógico y los pines de alimentación.

13 pines digitales numerados del D0 al D12:

1 pin analógico numerado A0

3 pines de 3,3V

1 pin de 5V (versión V3 2 pines 5V)

4 pines de tierra GND (versión V3 5 pines GND)

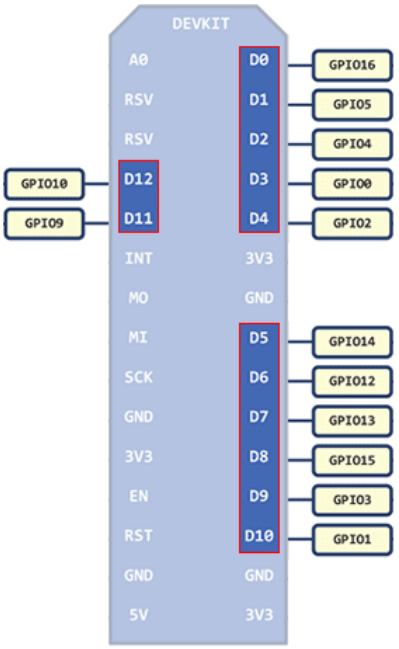
Con la versión NodeMCU V3 se tiene un pin más de 5V y de GND que ocuparían los pines nombrados como RSV.

Pines digitales de NodeMCU

Los pines digitales de NodeMCU van numerados del D0 al D12.

Hay dos nomenclaturas para nombrar los pines, los que aparecen en la placa escritos y en ocasiones el nombre asociado a cada Dx con GPIOx.

Ilustración 14 - Nomenclatura de pines



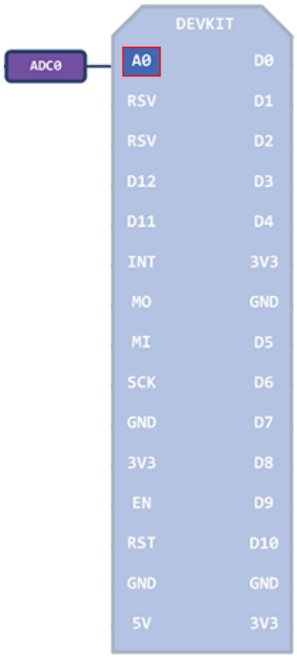
El GPIOx indica a qué pin del propio ESP8266 está conectado a esa patilla. Hay dos formas de acceder a ellos a través del código, esto puede generar cierta confusión.

Pin analógico de NodeMCU

Teniendo en cuenta que en los pines digitales solo se tienen dos estados LOW y HIGH, en el pin analógico se tienen un rango de valores. Este rango vendrá determinado por la resolución del conversor ADC (del inglés Analog Digital Converter) Conversor Analógico Digital.

El NodeMCU tiene solo un pin analógico que admite un rango de valores de 0 a 3,3V con una resolución de 10-bit.

Ilustración 15 - Pin analógico



Esto implica que dentro del código puede haber valores entre 0 y 1023 que se mapean con el voltaje entre 0 y 3,3V.

Pines de alimentación de NodeMCU

Los pines de alimentación tienen 2 funciones:

- Alimentar sensores y componentes (salida)
- Alimentar la propia placa (entrada)

El voltaje de operación de NodeMCU es de 3,3V y, por lo tanto, en principio no se puede alimentar ningún componente que necesitara 5V.

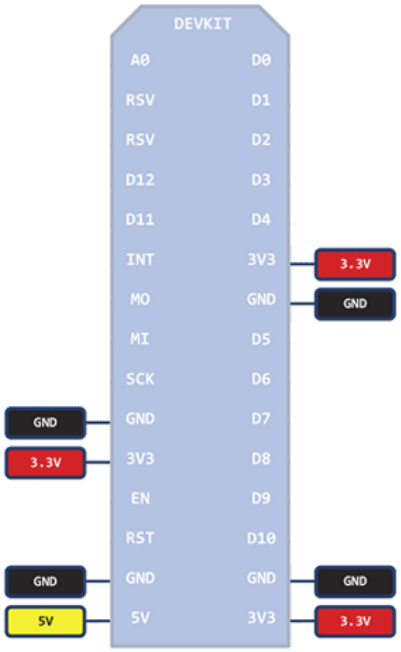
Sin embargo, cuando se alimenta a través del puerto USB con 5V, internamente tiene un regulador de voltaje que saca 3,3V y 5V. Los 3,3V se utilizan para alimentar el NodeMCU (electrónica) y para sacarlo por los 3 pines marcados con ese valor.

Los 5V se utilizan para alimentar otros componentes dentro de la placa y para sacarlos por el pin de 5V.

Además, por cualquiera de estos pines es posible suministrar el mismo voltaje permitiendo así alimentar a la placa además del puerto USB. Eso sí, si se alimenta con 3,3V por alguno de los pines marcados con ese valor, la salida de 5V ya no suministrará esos 5V.

Esto hay que tenerlo en cuenta dado que el proyecto hay sensores que se alimentan con 5V.

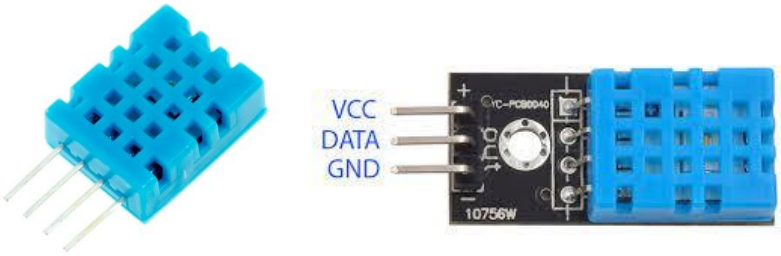
Ilustración 16 - Función de los pines



Sensor de temperatura y humedad relativa DHT11

El DHT11 es un sensor digital de temperatura y humedad relativa de bajo costo y fácil uso. Integra un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica). Utilizado en aplicaciones académicas relacionadas al control automático de temperatura, aire acondicionado, monitoreo ambiental en agricultura y más.

Ilustración 17 - Sensor DHT11



Utilizar el sensor DHT11 con las plataformas Arduino/Raspberry Pi/Nodemcu es muy sencillo tanto a nivel de software como hardware. A nivel de software se dispone de librerías para Arduino con soporte para el protocolo "Single bus". En cuanto al hardware, solo es necesario conectar el pin VCC de alimentación a 3-5V, el pin GND a Tierra (0V) y el pin de datos a un pin digital en nuestro Arduino. Si se desea conectar varios sensores DHT11 a un mismo Arduino, cada sensor debe tener su propio pin de datos. Quizá la única desventaja del sensor es que sólo se puede obtener nuevos datos cada 2 segundos. Cada sensor es calibrado en fabrica para obtener unos coeficientes de calibración grabados en su memoria OTP, asegurando alta estabilidad y fiabilidad a lo largo del tiempo. El protocolo de comunicación entre el sensor y el microcontrolador emplea un único hilo o cable, la distancia máxima recomendable de longitud de cable es de 20m, de preferencia se debe utilizar cable apantallado. Proteger el sensor de la luz directa del sol (radiación UV).

En comparación con el DHT22 y DHT21, este sensor es menos preciso, menos exacto y funciona en un rango más pequeño de temperatura / humedad, pero su empaque es más pequeño y de menor costo.

En el proyecto se va a utilizar el sensor que viene ya soldado a la placa electrónica, donde también se encuentran los componentes para un correcto funcionamiento del sensor.

ESPECIFICACIONES TÉCNICAS

Voltaje de Operación: 3V - 5V DC

Rango de medición de temperatura: 0 a 50 °C

Precisión de medición de temperatura: ±2.0 °C

Resolución Temperatura: 0.1°C

Rango de medición de humedad: 20% a 90% RH.

Precisión de medición de humedad: 5% RH.

Resolución Humedad: 1% RH

Tiempo de sensado: 1 seg.

Interface digital: Single-bus (bidireccional)

Modelo: DHT11

Dimensiones: 16*12*5 mm

Peso: 1 gr.

Carcasa de plástico celeste

PINES

1- Alimentación: +5V (VCC)

2- Datos (DATA)

3- No Usado (NC)

4- Tierra (GND)

8_ Sensor de Movimiento PIR HC-SR501

Sensor de movimiento o Sensor PIR (Passive Infra Red - sensor infrarrojo pasivo), mide la luz infrarroja (IR) radiada de los objetos situados en su campo de visión. Puede ajustar el tiempo de activación de la salida (3s hasta 300s) y la sensibilidad del sensor (3m hasta 7m).

Datos Técnicos

Voltaje de alimentación: 4,5 a 20V.

Niveles de salida: alto-3,3V, bajo – 0V.

Modos de disparo: L- no repite disparo, H- repite disparo

Tiempo de disparo ajustable: desde 3s a 300s.

Lente fresnel de 19 zonas, Ángulo <100°

Tamaño del lente de sensor: diametro-23mm.

Tamaño de tarjeta: 3,2 x 2,4 x 2,6 cm.

Los sensores infrarrojos pasivos (PIR) son dispositivos para la detección de movimiento. Son baratos, pequeños, de baja potencia, y fáciles de usar. Por esta razón son frecuentemente usados en juguetes, aplicaciones domóticas o sistemas de seguridad.

Los sensores PIR se basan en la medición de la radiación infrarroja. Todos los cuerpos (vivos o no) emiten una cierta cantidad de energía infrarroja, mayor cuanto mayor es su temperatura. Los dispositivos PIR disponen de un sensor piezo-eléctrico capaz de captar esta radiación y convertirla en una señal eléctrica.

En realidad, cada sensor está dividido en dos campos y se dispone de un circuito eléctrico que compensa ambas mediciones. Si ambos campos reciben la misma cantidad de infrarrojos la

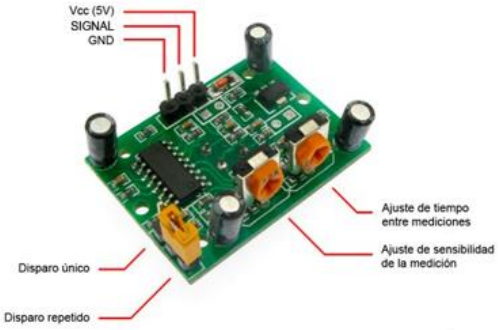
señal eléctrica resultante es nula. Por el contrario, si los dos campos realizan una medición diferente, se genera una señal eléctrica.

De esta forma, si un objeto atraviesa uno de los campos se genera una señal eléctrica diferencial, que es captada por el sensor, y se emite una señal digital.

El otro elemento restante para que todo funcione es la óptica del sensor. Básicamente es una cúpula de plástico formada por lentes de fresnel, que divide el espacio en zonas, y enfoca la radiación infrarroja a cada uno de los campos del PIR.

De esta manera, cada uno de los sensores capta un promedio de la radiación infrarroja del entorno. Cuando un objeto entra en el rango del sensor, alguna de las zonas marcadas por la óptica recibirá una cantidad distinta de radiación, que será captado por uno de los campos del sensor PIR, “disparando” la señal.

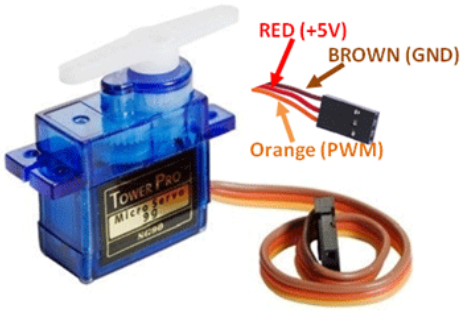
Ilustración 18 - Sensor PIR



Servomotor SG90

Es un pequeño actuador rotativo o bien motor que permite un control preciso en posición angular, este servomotor puede rotar de 0° hasta 180°, su voltaje de operación que va desde los 4.8 a 6 VDC. Este servo incluye 3 brazos y 3 tornillos, cuenta con un cable de hasta 25cm.

Ilustración 19 - Servomotor



Este tipo de servomotores son utilizado en gran variedad de proyectos de electrónica, robótica, carros de control remoto, aeronaves y más. Funcionan con la mayoría de tarjetas electrónicas de control con microcontroladores, como por ejemplo las tarjetas de Arduino, Nodemcu, Esp32, Pic´s y Raspberry Pi y otras.

El servo SG90 tiene un conector universal tipo “S” que encaja perfectamente en la mayoría controladores de servos por ejemplo el Controlador PCA9685 16 o el Probador de Servo 3CH ECS CCPM.

Este tipo de servo es ideal para las primeras experiencias de aprendizaje y prácticas con servos, ya que sus requerimientos de energía son bastante bajos y se permite alimentarlo con la misma fuente de alimentación que el circuito de control. Por ejemplo, si se conecta a una tarjeta Arduino, se puede alimentar durante las pruebas desde el puerto USB del PC sin mayor

problema, pero a veces se recomienda usar una fuente de alimentación independiente para el servomotor.

ESPECIFICACIONES Y CARACTERISTICAS

Modelo: SG90

Color: Azul

Tamaño: 22.8mm x 12.3mm x 22.5mm

Peso: 13 g

Grados / Angulo de Rotación Máximo: 0° a 180°

Engranajes: Nylon

Temperatura de trabajo: -30 a +60 Grados Celsius

7 microsegundos

Voltaje de funcionamiento: 4.8VDC a 6VDC. Recomendado 5VDC

Rojo =Alimentación (+)

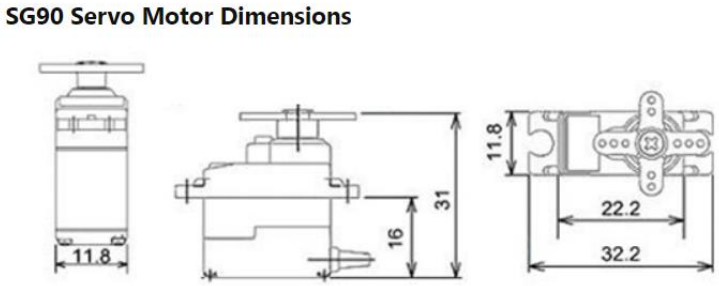
Café = Alimentación (–) o tierra

Naranja= Señal PWM

Rojo: VCC

Línea naranja: entrada de pulso

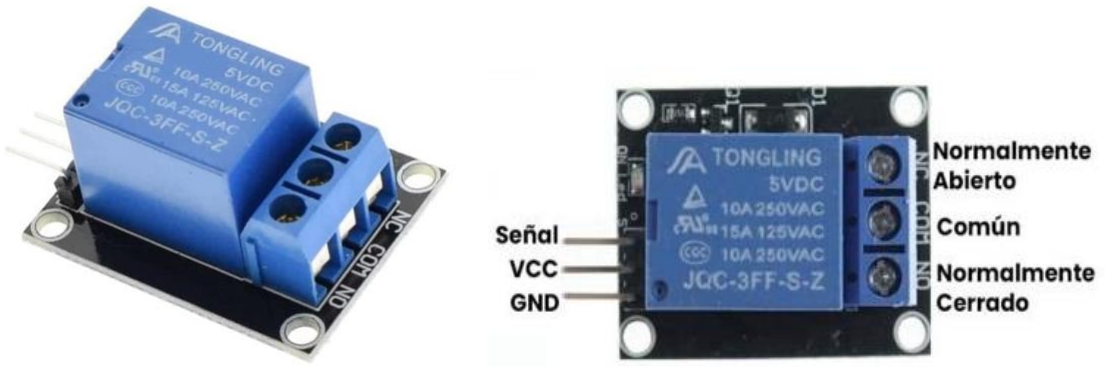
Ilustración 20 - Dimensiones Servomotor



Módulo Relee 5V KY-019

El Módulo Relee 5V KY-019 es un dispositivo electrónico que permite controlar dispositivos que trabajen en un máximo de *250V /15A* y *125V/10A* **AC** y *30V/10A* **DC** por medio de una señal de control de 5V, señalizando su activación por medio de un LED indicador en su PCB.

Ilustración 21 - Relee 5V KY-019



EL módulo KY-019 es utilizado en proyectos de IoT para controlar luces, contactos, electrodomésticos y otros dispositivos electrónicos. Comúnmente son usados en circuitos de control automático con una pequeña corriente de control y una gran corriente de operación.

ESPECIFICACIONES Y CARACTERÍSTICAS

Señal de control: 5V DC

Máxima AC: 250V /15A y 125V/10A AC

Máximo DC: 30V/10A DC

Tipo de contacto NC y NO

Dimensiones: 34 mmx 26 mmx 19 mm

Peso: 15 g

Programas de Instalaciones de Domótica

Programa Instalación Tipo A

```
/*------------------------------------------------------------------
 *
 *
 * CONEXIONES
 * SENSOR TEMPERATURA HUMEDAD - PIN D0 - GPIO16
 * SERVO                    - PIN D1 - GPIO5
 * SENSOR PIR               - PIN D5 - GPIO14
```

```
 * RELE                    - PIN D2 - GPIO4
 *
 *
 *
 */

#include <Servo.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h" // including the library of DHT11
#define DHTTYPE DHT11 // DHT 11 sensor usado
#define dht_dpin 0 //GPIO16 PIN D0
#define rele 4 //GPIO4 PIN D2
#define pirSensor 5 //GPIO14 PIN D5

Servo servo;
DHT dht(dht_dpin, DHTTYPE);
// DEFINICIONES---------- DATOS RED WIFI --------------------------
const char* ssid = "xxxx";
const char* password = "xxxxx";
float Setpoint=20.0;
// DEFINICIONES------------- MQTT ---------------------------------
const char* mqtt_server = "192.168.3.25";
```

```
WiFiClient espClient;
PubSubClient client(espClient);
//DEFINICIONES---------- VARIABLES GLOBALES----------
long lastMsg = 0;
unsigned long lastPir = 0; // Tempo en que se activo el PIR por ultima vez
char msg[150];
int value = 0;
boolean releState = LOW , ilAutomatica = LOW;
//*****************************************************************************
**********
//*********************************************SETUP*********************
**********
void setup(void){
//----------------- Declaracion de pines ------------------------------------
  pinMode(rele, OUTPUT);
  pinMode(pirSensor, INPUT_PULLUP);
  servo.attach(4); //Define donde esta conecatado el servo
  attachInterrupt(digitalPinToInterrupt(pirSensor),detectarMovimiento, RISING); // Set
Interrup PIR
  dht.begin();                    //Inicializa sensor de temperatura y humedad
  Serial.begin(115200);             //Inicializa el puerto serie
  setup_wifi();                    //Llama a la subrutina para inicializar la conexion WiFi
  client.setServer(mqtt_server, 1883);//Declara el servidor MQTT
```

```
  client.setCallback(callback);
  Serial.println("****************************************");
  Serial.println("Humedad y temperatura con DHT11");
  Serial.println("****************************************");
  delay(4000);//tiempo para mostrar el mensaje anterior
 }

 //************************************************************************
**********
 //*************************************SETUP
WIFI*********************************
 //Esta funcion realiza la conexion a la red WIFI

 void setup_wifi() {
  delay(10);
  // Conexion a la red WiFi
  Serial.println();
  Serial.print("Conectando a ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
   }

   Serial.println("");
   Serial.println("Conectado a WiFi");
   Serial.println("IP address: ");
   Serial.println(WiFi.localIP());
  }
  //**************************************************************************
**********
  //************************************RUTINA INTERRUPCION
PIR************************
  ICACHE_RAM_ATTR void detectarMovimiento() {
   Serial.println("Movimiento detectado");
   releState=digitalRead(rele);
   if(!releState && ilAutomatica){
    digitalWrite(rele, HIGH);
    lastPir = millis();
   }
  }
```

```
        //*************************************************************************
**********
        //************************* RECEPCION DE MENSAJES MQTT
*****************************
        void callback(String topic, byte* payload, unsigned int length) {
          String messageTemp;
          Serial.print("Mensaje MQTT entrante [");
          Serial.print(topic);
          Serial.print("] ");
          for (int i = 0; i < length; i++) {
            //Serial.print((char)payload[i]);
            messageTemp += (char)payload[i];
          }
          Serial.print(messageTemp);

          Serial.println();

           if(topic =="/frch/a2/spt"){
                Serial.print("Nueva Set point de temperatura ");
                Setpoint=messageTemp.toFloat();
                Serial.print(Setpoint);
           }
           else if(topic =="/frch/a2/laut"){
```

```
            if(messageTemp=="true"){
                Serial.print("RM - Iluminacion en automatico");
                ilAutomatica=HIGH;
                digitalWrite(rele,LOW);
            }
            else if(messageTemp=="false"){
                Serial.print("RM - Iluminacion en manual");
                ilAutomatica=LOW;
            }
          }

           else Serial.print("error en topic");
          Serial.println();
        }
        //********************************** Funcion para reconectar el cliente al servidor MQTT **********************
        void reconnect() {
          // Loop until we're reconnected
          while (!client.connected()) {
            Serial.print("Attempting MQTT connection...");
            // Attempt to connect
            if (client.connect("ESP8266Client")) {
              Serial.println("Conectado al Broker");
```

```
      // Once connected, publish an announcement...
      client.publish("/frch/a2", "Enviando el primer mensaje");
      // ... and resubscribe
      client.subscribe("/frch/a2/spt");
      client.subscribe("/frch/a2/laut");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
//*************************************************************************
****************************************
//****************************************Funcion principal
LOOP*************************************************

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  int cierre=(90*(t-(Setpoint+0.5)))+180; // calcula el grado de apertura intermedio
```

```
  int apservo=0;
  long unsigned int tiempoActual;
  char msgTemp[256],temp[5];
  //chequeo de tiempo
  tiempoActual = millis();
  releState=digitalRead(rele);
/***************Control de apertura*********/

  if (t>(Setpoint+0.5)) {
        servo.write(180); // abre
        delay(1000); // tiempo entre lecturas
        }
  if (t<(Setpoint-0.5)) {
        servo.write(90); // abre
        delay(1000); // tiempo entre lecturas
        }
  if ((t>(Setpoint-0.5)) && (t<(Setpoint+0.5))){
        servo.write(cierre); // cierra en punto intermedio
        delay(1000);
        }
  apservo=180 - servo.read();
  Serial.print("Humedad = ");
  Serial.print(h);
```

```
  Serial.print(" % ");
  Serial.print("|Temperatura = ");
  Serial.print(t);
  Serial.print(" °C ");
  Serial.print("|Apertura=");
  Serial.print(apservo);
  Serial.print("°");
  Serial.print("|SetPoint=");
  Serial.print(Setpoint);
  Serial.print("|Ilum Auto=");
  Serial.println(releState);
  Serial.println("----------------------------------------");

  long now = millis();        //---Milisegundos actuales
/*******************MQTT******************/
  if (!client.connected()) {  //Si se desconecto del brocker volver a conectar
    reconnect();
  }
  client.loop();

  if (now - lastMsg > 2000) {
    lastMsg = now;
    dtostrf(t,2,2,temp);
```

```
            snprintf (msg, 75, "{\"t\":%s,\"h\":%i,\"ap\":%i,\"sp\":%i}",
temp,(int)h,apservo,(int)Setpoint);
            Serial.print("Publish message: ");
            Serial.println(msg);
            client.publish("/frch/a2", msg);
          }
          if (!ilAutomatica && !releState){
              Serial.println("L - Encendido luces manual");
              digitalWrite(rele,HIGH);
          }
          else if (ilAutomatica && (now - lastPir > 120000)&& releState) {
              digitalWrite(rele,LOW);
              Serial.println("L - Se apagan las luces");
          }
        }
```

Programa Instalacion Tipo B

```
/*-----------------------------------------------------------------------
 *
 *
 * CONEXIONES
 * SENSOR TEMPERATURA HUMEDAD - PIN D0 - GPIO16
 * SERVO                      - PIN D1 - GPIO5
 * SENSOR PIR 1               - PIN D5 - GPIO14
 * SENSOR PIR 2               - PIN D6 - GPIO12
 * RELE                       - PIN D2 - GPIO4
 *
 *
 *
 */


#include <Servo.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h" // including the library of DHT11
#define DHTTYPE DHT11 // DHT 11 sensor usado
#define dht_dpin 0 //GPIO16 PIN D0
#define rele 4 //GPIO4 PIN D2
```

```
#define pirSensor1 14 //GPIO14 PIN D5
#define pirSensor2 12 //GPIO12 PIN D6


Servo servo;
DHT dht(dht_dpin, DHTTYPE);
// DEFINICIONES---------- DATOS RED WIFI -------------------------
const char* ssid = "xxxx";
const char* password = "xxxxx";
float Setpoint=20.0;
// DEFINICIONES------------- MQTT --------------------------------
const char* mqtt_server = "192.168.3.25";
WiFiClient espClient;
PubSubClient client(espClient);
//DEFINICIONES------------- VARIABLES GLOBALES--------------------
long lastMsg = 0;
unsigned long lastPir = 0; // Tempo en que se activo el PIR por ultima vez
char msg[150];
int value = 0;
boolean releState = LOW , ilAutomatica = LOW;
//*****************************************************************************
**********
//*********************************************SETUP*********************
**********
```

```
void setup(void){
//----------------- Declaracion de pines ------------------------------------
  pinMode(rele, OUTPUT);
  pinMode(pirSensor1, INPUT_PULLUP);
  pinMode(pirSensor2, INPUT_PULLUP);
  servo.attach(5); //Define donde esta conecatado el servo
  attachInterrupt(digitalPinToInterrupt(pirSensor1),detectarMovimiento, RISING); // Set
Interrup PIR
  attachInterrupt(digitalPinToInterrupt(pirSensor2),detectarMovimiento, RISING); // Set
Interrup PIR
  dht.begin();                    //Inicializa sensor de temperatura y humedad
  Serial.begin(115200);           //Inicializa el puerto serie
  setup_wifi();                   //Llama a la subrutina para inicializar la conexion WiFi
  client.setServer(mqtt_server, 1883);//Declara el servidor MQTT
  client.setCallback(callback);
  Serial.println("*****************************************");
  Serial.println("Humedad y temperatura con DHT11");
  Serial.println("*****************************************");
  delay(4000);//tiempo para mostrar el mensaje anterior
}
```

```
        //*************************************************************************
**********
        //****************************************SETUP
WIFI*********************************
        //Esta funcion realiza la conexion a la red WIFI

        void setup_wifi() {
          delay(10);
          // Conexion a la red WiFi
          Serial.println();
          Serial.print("Conectando a ");
          Serial.println(ssid);

          WiFi.begin(ssid, password);

          while (WiFi.status() != WL_CONNECTED) {
            delay(500);
            Serial.print(".");
          }

          Serial.println("");
          Serial.println("Conectado a WiFi");
          Serial.println("IP address: ");
```

```
    Serial.println(WiFi.localIP());
  }
  //*************************************************************************
**********
  //*************************************RUTINA INTERRUPCION
PIR**********************
  ICACHE_RAM_ATTR void detectarMovimiento() {
    Serial.println("Movimiento detectado");
    releState=digitalRead(rele);
    if(!releState && ilAutomatica){
      digitalWrite(rele, HIGH);
      lastPir = millis();
    }
  }

  //*************************************************************************
**********
  //************************** RECEPCION DE MENSAJES MQTT
*****************************
  void callback(String topic, byte* payload, unsigned int length) {
    String messageTemp;
    Serial.print("Mensaje MQTT entrante [");
    Serial.print(topic);
```

```
Serial.print("] ");
for (int i = 0; i < length; i++) {
  //Serial.print((char)payload[i]);
  messageTemp += (char)payload[i];
}
Serial.print(messageTemp);

Serial.println();

 if(topic =="/frch/a2/spt"){
      Serial.print("Nueva Set point de temperatura ");
      Setpoint=messageTemp.toFloat();
      Serial.print(Setpoint);
 }
 else if(topic =="/frch/a2/laut"){
    if(messageTemp=="true"){
        Serial.print("RM - Iluminacion en automatico");
        ilAutomatica=HIGH;
        digitalWrite(rele,LOW);
    }
    else if(messageTemp=="false"){
        Serial.print("RM - Iluminacion en manual");
        ilAutomatica=LOW;
```

```
    }
   }

    else Serial.print("error en topic");
   Serial.println();
  }
  //********************************** Funcion para reconectar el cliente al servidor MQTT **********************
  void reconnect() {
   // Loop until we're reconnected
   while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
     Serial.println("Conectado al Broker");
     // Once connected, publish an announcement...
     client.publish("/frch/a2", "Enviando el primer mensaje");
     // ... and resubscribe
     client.subscribe("/frch/a2/spt");
     client.subscribe("/frch/a2/laut");
    } else {
     Serial.print("failed, rc=");
     Serial.print(client.state());
```

```
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
//*************************************************************************
*************************************
//*************************************Funcion principal
LOOP*************************************************

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  int cierre=(90*(t-(Setpoint+0.5)))+180; // calcula el grado de apertura intermedio
  int apservo=0;
  long unsigned int tiempoActual;
  char msgTemp[256],temp[5];
  //chequeo de tiempo
  tiempoActual = millis();
  releState=digitalRead(rele);
/**************Control de apertura*********/
```

```
if (t>(Setpoint+0.5)) {
        servo.write(180); // abre
        delay(1000); // tiempo entre lecturas
        }
if (t<(Setpoint-0.5)) {
        servo.write(90); // abre
        delay(1000); // tiempo entre lecturas
        }
if ((t>(Setpoint-0.5)) && (t<(Setpoint+0.5))){
        servo.write(cierre); // cierra en punto intermedio
        delay(1000);
        }
apservo=180 - servo.read();
Serial.print("Humedad = ");
Serial.print(h);
Serial.print(" % ");
Serial.print("|Temperatura = ");
Serial.print(t);
Serial.print(" °C ");
Serial.print("|Apertura=");
Serial.print(apservo);
Serial.print("°");
Serial.print("|SetPoint=");
```

```
            Serial.print(Setpoint);
            Serial.print("|Ilum Auto=");
            Serial.println(releState);
            Serial.println("----------------------------------------");

            long now = millis();        //---Milisegundos actuales
          /*******************MQTT******************/
            if (!client.connected()) {  //Si se desconecto del brocker volver a conectar
              reconnect();
            }
            client.loop();

            if (now - lastMsg > 2000) {
              lastMsg = now;
              dtostrf(t,2,2,temp);
              snprintf (msg, 75, "{\"t\":%s,\"h\":%i,\"ap\":%i,\"sp\":%i}",
temp,(int)h,apservo,(int)Setpoint);
              Serial.print("Publish message: ");
              Serial.println(msg);
              client.publish("/frch/a2", msg);
            }
           if (!ilAutomatica && !releState){
               Serial.println("L - Encendido luces manual");
```

```
    digitalWrite(rele,HIGH);
  }
  else if (ilAutomatica && (now - lastPir > 120000)&& releState) {
    digitalWrite(rele,LOW);
    Serial.println("L - Se apagan las luces");
  }
 }
```

Programa Instalacion Tipo C

```
/*------------------------------------------------------------------------
 *
 *
 * CONEXIONES
 * SENSOR TEMPERATURA HUMEDAD - PIN D0 - GPIO16
 * SERVO 1                - PIN D1 - GPIO5
 * SERVO 2                - PIN D7 - GPIO13
 * SENSOR PIR 1             - PIN D5 - GPIO14
 * SENSOR PIR 2             - PIN D6 - GPIO12
 * RELE                 - PIN D2 - GPIO4
 *
 *
```

```
 *

 */


#include <Servo.h>

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include "DHT.h" // including the library of DHT11

#define DHTTYPE DHT11 // DHT 11 sensor usado

#define dht_dpin 0 //GPIO16 PIN D0

#define rele 4 //GPIO4 PIN D2

#define pirSensor1 14 //GPIO14 PIN D5

#define pirSensor2 12 //GPIO12 PIN D6


Servo servo1;

Servo servo2;

DHT dht(dht_dpin, DHTTYPE);

// DEFINICIONES---------- DATOS RED WIFI -------------------------

const char* ssid = "xxxx";

const char* password = "xxxxx";

float Setpoint=20.0;

// DEFINICIONES------------- MQTT --------------------------------

const char* mqtt_server = "192.168.3.25";

WiFiClient espClient;
```

```
PubSubClient client(espClient);
//DEFINICIONES---------- VARIABLES GLOBALES----------
long lastMsg = 0;
unsigned long lastPir = 0; // Tempo en que se activo el PIR por ultima vez
char msg[150];
int value = 0;
boolean releState = LOW , ilAutomatica = LOW;
//**********************************************************************
**********
//*********************************************SETUP*********************
**********

void setup(void){
//----------------- Declaracion de pines ------------------------------------
  pinMode(rele, OUTPUT);
  pinMode(pirSensor1, INPUT_PULLUP);
  pinMode(pirSensor2, INPUT_PULLUP);
  servo1.attach(5); //Define donde esta conecatado el servo1
  servo2.attach(13); //Define donde esta conecatado el servo2
  attachInterrupt(digitalPinToInterrupt(pirSensor1),detectarMovimiento, RISING); // Set
Interrup PIR
  attachInterrupt(digitalPinToInterrupt(pirSensor2),detectarMovimiento, RISING); // Set
Interrup PIR
```

```
  dht.begin();                    //Inicializa sensor de temperatura y humedad
  Serial.begin(115200);           //Inicializa el puerto serie
  setup_wifi();                   //Llama a la subrutina para inicializar la conexion WiFi
  client.setServer(mqtt_server, 1883);//Declara el servidor MQTT
  client.setCallback(callback);
  Serial.println("*****************************************");
  Serial.println("Humedad y temperatura con DHT11");
  Serial.println("*****************************************");
  delay(4000);//tiempo para mostrar el mensaje anterior
 }

 //************************************************************************
**********
 //*****************************************SETUP
WIFI***********************************
 //Esta funcion realiza la conexion a la red WIFI

 void setup_wifi() {
  delay(10);
  // Conexion a la red WiFi
  Serial.println();
  Serial.print("Conectando a ");
  Serial.println(ssid);
```

```
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("Conectado a WiFi");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
//**************************************************************************
**********
//*************************************RUTINA INTERRUPCION
PIR************************
ICACHE_RAM_ATTR void detectarMovimiento() {
  Serial.println("Movimiento detectado");
  releState=digitalRead(rele);
  if(!releState && ilAutomatica){
    digitalWrite(rele, HIGH);
    lastPir = millis();
```

```
  }
 }

 //*************************************************************************
**********
 //************************* RECEPCION DE MENSAJES MQTT
*****************************
 void callback(String topic, byte* payload, unsigned int length) {
  String messageTemp;
  Serial.print("Mensaje MQTT entrante [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
   //Serial.print((char)payload[i]);
   messageTemp += (char)payload[i];
  }
  Serial.print(messageTemp);

  Serial.println();

  if(topic =="/frch/a2/spt"){
      Serial.print("Nueva Set point de temperatura ");
      Setpoint=messageTemp.toFloat();
```

```
            Serial.print(Setpoint);
          }
          else if(topic =="/frch/a2/laut"){
            if(messageTemp=="true"){
                Serial.print("RM - Iluminacion en automatico");
                ilAutomatica=HIGH;
                digitalWrite(rele,LOW);
            }
            else if(messageTemp=="false"){
                Serial.print("RM - Iluminacion en manual");
                ilAutomatica=LOW;
            }
          }
          else Serial.print("error en topic");
         Serial.println();
        }
        //********************************** Funcion para reconectar el cliente al servidor
MQTT **********************
        void reconnect() {
         // Loop until we're reconnected
         while (!client.connected()) {
           Serial.print("Attempting MQTT connection...");
           // Attempt to connect
```

```
      if (client.connect("ESP8266Client")) {
        Serial.println("Conectado al Broker");
        // Once connected, publish an announcement...
        client.publish("/frch/a2", "Enviando el primer mensaje");
        // ... and resubscribe
        client.subscribe("/frch/a2/spt");
        client.subscribe("/frch/a2/laut");
      } else {
        Serial.print("failed, rc=");
        Serial.print(client.state());
        Serial.println(" try again in 5 seconds");
        // Wait 5 seconds before retrying
        delay(5000);
      }
    }
  }
  //*************************************************************************
**************************************
  //***************************************Funcion principal
LOOP*************************************************

  void loop() {
    float h = dht.readHumidity();
```

```
  float t = dht.readTemperature();
  int cierre=(90*(t-(Setpoint+0.5)))+180; // calcula el grado de apertura intermedio
  int apservo=0;
  long unsigned int tiempoActual;
  char msgTemp[256],temp[5];
  //chequeo de tiempo
  tiempoActual = millis();
  releState=digitalRead(rele);
/***************Control de apertura**********/

  if (t>(Setpoint+0.5)) {
        servo1.write(180); // abre
        servo2.write(180);
        delay(1000); // tiempo entre lecturas
        }
  if (t<(Setpoint-0.5)) {
        servo1.write(90); // abre
        servo2.write(90); // abre
        delay(1000); // tiempo entre lecturas
        }
  if ((t>(Setpoint-0.5)) && (t<(Setpoint+0.5))){
        servo1.write(cierre); // cierra en punto intermedio
        servo2.write(cierre);
```

```
        delay(1000);
        }
  apservo=180 - servo.read();
  Serial.print("Humedad = ");
  Serial.print(h);
  Serial.print(" % ");
  Serial.print("|Temperatura = ");
  Serial.print(t);
  Serial.print(" °C ");
  Serial.print("|Apertura=");
  Serial.print(apservo);
  Serial.print("°");
  Serial.print("|SetPoint=");
  Serial.print(Setpoint);
  Serial.print("|Ilum Auto=");
  Serial.println(releState);
  Serial.println("----------------------------------------");

  long now = millis();        //---Milisegundos actuales
/*******************MQTT******************/
  if (!client.connected()) {  //Si se desconecto del brocker volver a conectar
    reconnect();
  }
```

```
 client.loop();


 if (now - lastMsg > 2000) {
   lastMsg = now;
   dtostrf(t,2,2,temp);
   snprintf (msg, 75, "{\"t\":%s,\"h\":%i,\"ap\":%i,\"sp\":%i}",
temp,(int)h,apservo,(int)Setpoint);
   Serial.print("Publish message: ");
   Serial.println(msg);
   client.publish("/frch/a2", msg);
 }
 if (!ilAutomatica && !releState){
     Serial.println("L - Encendido luces manual");
     digitalWrite(rele,HIGH);
 }
 else if (ilAutomatica && (now - lastPir > 120000)&& releState) {
     digitalWrite(rele,LOW);
     Serial.println("L - Se apagan las luces");
 }
}
```

Programa Instalacion Tipo D

```
/*-------------------------------------------------------------------------
 *
 *
 * CONEXIONES
 * SENSOR TEMPERATURA HUMEDAD 1 - PIN D0 - GPIO16
 * SENSOR TEMPERATURA HUMEDAD 2 - PIN D3 - GPIO0
 * SERVO 1                 - PIN D1 - GPIO5
 * SERVO 2                 - PIN D7 - GPIO13
 * SENSOR PIR 1              - PIN D5 - GPIO14
 * SENSOR PIR 2              - PIN D6 - GPIO12
 * RELE                  - PIN D2 - GPIO4
 *
 *
 *
 */


#include <Servo.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h" // including the library of DHT11
#define DHTTYPE DHT11 // DHT 11 sensor usado
```

```
#define dht_dpin 0 //GPIO16 PIN D0
#define dth1_dpin 1 //GPIO0 PIN D3  - Se define el pin D1 como el sensor de temperatura 1
#define rele 4 //GPIO4 PIN D2
#define pirSensor1 14 //GPIO14 PIN D5
#define pirSensor2 12 //GPIO12 PIN D6


Servo servo1;
Servo servo2;
DHT dht(dht_dpin, DHTTYPE);
DHT dht1(dth1_dpin, DHTTYPE); // se agrego la configuracion del segundo sensor de temperatura
// DEFINICIONES---------- DATOS RED WIFI -------------------------
const char* ssid = "xxxx";
const char* password = "xxxxx";
float Setpoint=20.0;
// DEFINICIONES------------- MQTT --------------------------------
const char* mqtt_server = "192.168.3.25";
WiFiClient espClient;
PubSubClient client(espClient);
//DEFINICIONES------------- VARIABLES GLOBALES--------------------
long lastMsg = 0;
unsigned long lastPir = 0; // Tempo en que se activo el PIR por ultima vez
```

```
char msg[150];
int value = 0;
boolean releState = LOW , ilAutomatica = LOW;
//**********************************************************************
**********
//*********************************************SETUP*********************
**********

void setup(void){
//----------------- Declaracion de pines ------------------------------------
  pinMode(rele, OUTPUT);
  pinMode(pirSensor1, INPUT_PULLUP);
  pinMode(pirSensor2, INPUT_PULLUP);
  servo1.attach(5); //Define donde esta conecatado el servo1
  servo2.attach(13); //Define donde esta conecatado el servo2
  attachInterrupt(digitalPinToInterrupt(pirSensor1),detectarMovimiento, RISING); // Set
Interrup PIR
  attachInterrupt(digitalPinToInterrupt(pirSensor2),detectarMovimiento, RISING); // Set
Interrup PIR
  dht.begin();                  //Inicializa sensor de temperatura y humedad 1
  dht1.begin();                  //Inicializa sensor de temperatura y humedad 2
  Serial.begin(115200);             //Inicializa el puerto serie
  setup_wifi();                  //Llama a la subrutina para inicializar la conexion WiFi
```

```
  client.setServer(mqtt_server, 1883);//Declara el servidor MQTT
  client.setCallback(callback);
  Serial.println("*****************************************");
  Serial.println("Humedad y temperatura con DHT11");
  Serial.println("*****************************************");
  delay(4000);//tiempo para mostrar el mensaje anterior
 }

 //************************************************************************
**********
 //*************************************SETUP
WIFI*********************************
 //Esta funcion realiza la conexion a la red WIFI

 void setup_wifi() {
  delay(10);
  // Conexion a la red WiFi
  Serial.println();
  Serial.print("Conectando a ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("Conectado a WiFi");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
 }
 //************************************************************************
**********
 //*************************************RUTINA INTERRUPCION
PIR***********************
 ICACHE_RAM_ATTR void detectarMovimiento() {
  Serial.println("Movimiento detectado");
  releState=digitalRead(rele);
  if(!releState && ilAutomatica){
    digitalWrite(rele, HIGH);
    lastPir = millis();
  }
 }
```

```
        //*************************************************************************
**********
        //************************* RECEPCION DE MENSAJES MQTT
*****************************
        void callback(String topic, byte* payload, unsigned int length) {
          String messageTemp;
          Serial.print("Mensaje MQTT entrante [");
          Serial.print(topic);
          Serial.print("] ");
          for (int i = 0; i < length; i++) {
            //Serial.print((char)payload[i]);
            messageTemp += (char)payload[i];
          }
          Serial.print(messageTemp);

          Serial.println();

           if(topic =="/frch/a2/spt"){
                Serial.print("Nueva Set point de temperatura ");
                Setpoint=messageTemp.toFloat();
                Serial.print(Setpoint);
           }
           else if(topic =="/frch/a2/laut"){
```

```
            if(messageTemp=="true"){
                Serial.print("RM - Iluminacion en automatico");
                ilAutomatica=HIGH;
                digitalWrite(rele,LOW);
            }
            else if(messageTemp=="false"){
                Serial.print("RM - Iluminacion en manual");
                ilAutomatica=LOW;
            }
          }

           else Serial.print("error en topic");
          Serial.println();
        }
        //********************************** Funcion para reconectar el cliente al servidor MQTT **********************
        void reconnect() {
          // Loop until we're reconnected
          while (!client.connected()) {
            Serial.print("Attempting MQTT connection...");
            // Attempt to connect
            if (client.connect("ESP8266Client")) {
              Serial.println("Conectado al Broker");
```

```
            // Once connected, publish an announcement...
            client.publish("/frch/a2", "Enviando el primer mensaje");
            // ... and resubscribe
            client.subscribe("/frch/a2/spt");
            client.subscribe("/frch/a2/laut");
          } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
          }
        }
      }
      //*************************************************************************
*****************************************
      //****************************************Funcion principal
LOOP*************************************************

      void loop() {
        float h,t;
        float h1 = dht1.readHumidity();
        float t1 = dht1.readTemperature();
```

```
  float h2 = dht2.readHumidity();
  float t2 = dht2.readTemperature();
  h=(h1+h2)/2; //Promedio de humedad de los 2 sensores
  t=(t1+t2)/2; //Promedio de temperatura de los 2 sensores
  int cierre=(90*(t-(Setpoint+0.5)))+180; // calcula el grado de apertura intermedio
  int apservo=0;
  long unsigned int tiempoActual;
  char msgTemp[256],temp[5];
  //chequeo de tiempo
  tiempoActual = millis();
  releState=digitalRead(rele);
/***************Control de apertura**********/

  if (t>(Setpoint+0.5)) {
        servo1.write(180); // abre
        servo2.write(180);
        delay(1000); // tiempo entre lecturas
        }
  if (t<(Setpoint-0.5)) {
        servo1.write(90); // abre
        servo2.write(90); // abre
        delay(1000); // tiempo entre lecturas
        }
```

```
  if ((t>(Setpoint-0.5)) && (t<(Setpoint+0.5))){
        servo1.write(cierre); // cierra en punto intermedio
        servo2.write(cierre);
        delay(1000);
        }
  apservo=180 - servo1.read();
  Serial.print("Humedad = ");
  Serial.print(h);
  Serial.print(" % ");
  Serial.print("|Temperatura = ");
  Serial.print(t);
  Serial.print(" °C ");
  Serial.print("|Apertura=");
  Serial.print(apservo);
  Serial.print("°");
  Serial.print("|SetPoint=");
  Serial.print(Setpoint);
  Serial.print("|Ilum Auto=");
  Serial.println(releState);
  Serial.println("----------------------------------------");

  long now = millis();        //---Milisegundos actuales
/*******************MQTT******************/
```

```
 if (!client.connected()) {  //Si se desconecto del brocker volver a conectar
   reconnect();
 }
 client.loop();

 if (now - lastMsg > 2000) {
   lastMsg = now;
   dtostrf(t,2,2,temp);
   snprintf (msg, 75, "{\"t\":%s,\"h\":%i,\"ap\":%i,\"sp\":%i}",
temp,(int)h,apservo,(int)Setpoint);
   Serial.print("Publish message: ");
   Serial.println(msg);
   client.publish("/frch/a2", msg);
 }
 if (!ilAutomatica && !releState){
     Serial.println("L - Encendido luces manual");
     digitalWrite(rele,HIGH);
 }
 else if (ilAutomatica && (now - lastPir > 120000)&& releState) {
     digitalWrite(rele,LOW);
     Serial.println("L - Se apagan las luces");
 }
}
```

Programa Instalación Tipo E

```
/*-----------------------------------------------------------------------
 *
 *
 * CONEXIONES
 * SENSOR TEMPERATURA HUMEDAD 1 - PIN D0 - GPIO16
 * SENSOR TEMPERATURA HUMEDAD 2 - PIN D1 - GPIO5
 * SERVO 1                - PIN D2 - GPIO4
 * SERVO 2                - PIN D3 - GPIO0
 * SERVO 3                - PIN D4 - GPIO2
 * SENSOR PIR 1             - PIN D6 - GPIO12
 * SENSOR PIR 2             - PIN D7 - GPIO13
 * SENSOR PIR 3             - PIN D8 - GPIO15
 * RELE                 - PIN D5 - GPIO14
 *
 *
 *
 */


#include <Servo.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
```

```
#include "DHT.h" // including the library of DHT11
#define DHTTYPE DHT11 // DHT 11 sensor usado
#define dht_dpin 0 //GPIO16 PIN D0
#define dth1_dpin 1 //GPIO5 PIN D1  - Se define el pin D1 como el sensor de temperatura 1
#define rele 14 //GPIO14 PIN D5
#define pirSensor1 12 //GPIO12 PIN D6
#define pirSensor2 13 //GPIO13 PIN D7
#define pirSensor3 15 //GPIO15 PIN D8

Servo servo1;
Servo servo2;
Servo servo3;
DHT dht(dht_dpin, DHTTYPE);
DHT dht1(dth1_dpin, DHTTYPE); // se agrego la configuracion del segundo sensor de temperatura
// DEFINICIONES---------- DATOS RED WIFI -------------------------
const char* ssid = "xxxx";
const char* password = "xxxxx";
float Setpoint=20.0;
// DEFINICIONES------------- MQTT --------------------------------
const char* mqtt_server = "192.168.3.25";
WiFiClient espClient;
```

```
PubSubClient client(espClient);
//DEFINICIONES---------- VARIABLES GLOBALES----------
long lastMsg = 0;
unsigned long lastPir = 0; // Tempo en que se activo el PIR por ultima vez
char msg[150];
int value = 0;
boolean releState = LOW , ilAutomatica = LOW;
//**********************************************************************
**********
//*********************************************SETUP*********************
**********

void setup(void){
//----------------- Declaracion de pines -------------------------------------
  pinMode(rele, OUTPUT);
  pinMode(pirSensor1, INPUT_PULLUP);
  pinMode(pirSensor2, INPUT_PULLUP);
  pinMode(pirSensor3, INPUT_PULLUP);
  servo1.attach(4); //Define donde esta conecatado el servo1
  servo2.attach(0); //Define donde esta conecatado el servo2
  servo3.attach(2); //Define donde esta conecatado el servo3
  attachInterrupt(digitalPinToInterrupt(pirSensor1),detectarMovimiento, RISING); // Set
Interrup PIR
```

```
    attachInterrupt(digitalPinToInterrupt(pirSensor2),detectarMovimiento, RISING); // Set
Interrup PIR
    attachInterrupt(digitalPinToInterrupt(pirSensor3),detectarMovimiento, RISING); // Set
Interrup PIR
    dht.begin();                  //Inicializa sensor de temperatura y humedad 1
    dht1.begin();                 //Inicializa sensor de temperatura y humedad 2
    Serial.begin(115200);         //Inicializa el puerto serie
    setup_wifi();                 //Llama a la subrutina para inicializar la conexion WiFi
    client.setServer(mqtt_server, 1883);//Declara el servidor MQTT
    client.setCallback(callback);
    Serial.println("****************************************");
    Serial.println("Humedad y temperatura con DHT11");
    Serial.println("****************************************");
    delay(4000);//tiempo para mostrar el mensaje anterior
  }

  //************************************************************************
**********
  //*****************************************SETUP
WIFI*********************************
  //Esta funcion realiza la conexion a la red WIFI

  void setup_wifi() {
```

```
  delay(10);
  // Conexion a la red WiFi
  Serial.println();
  Serial.print("Conectando a ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("Conectado a WiFi");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
//*****************************************************************************
**********
//*************************************RUTINA INTERRUPCION
PIR***********************
ICACHE_RAM_ATTR void detectarMovimiento() {
```

```
    Serial.println("Movimiento detectado");
    releState=digitalRead(rele);
    if(!releState && ilAutomatica){
      digitalWrite(rele, HIGH);
      lastPir = millis();
    }
  }

  //*****************************************************************************
**********
  //************************* RECEPCION DE MENSAJES MQTT
*****************************
  void callback(String topic, byte* payload, unsigned int length) {
    String messageTemp;
    Serial.print("Mensaje MQTT entrante [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
      messageTemp += (char)payload[i];
    }
    Serial.print(messageTemp);

    Serial.println();
```

```
  if(topic =="/frch/a2/spt"){
      Serial.print("Nueva Set point de temperatura ");
      Setpoint=messageTemp.toFloat();
      Serial.print(Setpoint);
  }
  else if(topic =="/frch/a2/laut"){
    if(messageTemp=="true"){
        Serial.print("RM - Iluminacion en automatico");
        ilAutomatica=HIGH;
        digitalWrite(rele,LOW);
    }
    else if(messageTemp=="false"){
        Serial.print("RM - Iluminacion en manual");
        ilAutomatica=LOW;
    }
  }

  else Serial.print("error en topic");
 Serial.println();
}
//********************************** Funcion para reconectar el cliente al servidor MQTT **********************
void reconnect() {
```

```
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
      Serial.println("Conectado al Broker");
      // Once connected, publish an announcement...
      client.publish("/frch/a2", "Enviando el primer mensaje");
      // ... and resubscribe
      client.subscribe("/frch/a2/spt");
      client.subscribe("/frch/a2/laut");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
//*****************************************************************************
*****************************************
```

```
//****************************************Funcion principal
LOOP*************************************************

void loop() {
  float h1 = dht.readHumidity();
  float t1 = dht.readTemperature();
  float h2 = dht1.readHumidity();
  float t2 = dht1.readTemperature();
  float h=(h1+h2)/2;
  float t=(t1+t2)/2;
  int cierre=(90*(t-(Setpoint+0.5)))+180; // calcula el grado de apertura intermedio
  int apservo1=0;
  long unsigned int tiempoActual;
  char msgTemp[256],temp[5];
  //chequeo de tiempo
  tiempoActual = millis();
  releState=digitalRead(rele);
/**************Control de apertura*********/

  if (t>(Setpoint+0.5)) {
        servo1.write(180); // abre
        servo2.write(180);
        servo3.write(180);
```

```
        delay(1000); // tiempo entre lecturas
        }
if (t<(Setpoint-0.5)) {
        servo1.write(90); // abre
        servo2.write(90);
        servo3.write(90);
        delay(1000); // tiempo entre lecturas
        }
if ((t>(Setpoint-0.5)) && (t<(Setpoint+0.5))){
        servo1.write(cierre); // cierra en punto intermedio
        servo2.write(cierre);
        servo3.write(cierre);
        delay(1000);
        }
apservo1=180 - servo1.read();
Serial.print("Humedad = ");
Serial.print(h);
Serial.print(" % ");
Serial.print("|Temperatura = ");
Serial.print(t);
Serial.print(" °C ");
Serial.print("|Apertura=");
Serial.print(apservo1);
```

```
    Serial.print("°");
    Serial.print("|SetPoint=");
    Serial.print(Setpoint);
    Serial.print("|Ilum Auto=");
    Serial.println(releState);
    Serial.println("----------------------------------------");

    long now = millis();        //---Milisegundos actuales
  /*******************MQTT******************/
    if (!client.connected()) {  //Si se desconecto del brocker volver a conectar
      reconnect();
    }
    client.loop();

    if (now - lastMsg > 2000) {
      lastMsg = now;
      dtostrf(t,2,2,temp);
      snprintf (msg, 75, "{\"t\":%s,\"h\":%i,\"ap\":%i,\"sp\":%i}",
temp,(int)h,apservo,(int)Setpoint);
      Serial.print("Publish message: ");
      Serial.println(msg);
      client.publish("/frch/a2", msg);
    }
```

```
 if (!ilAutomatica && !releState){
     Serial.println("L - Encendido luces manual");
     digitalWrite(rele,HIGH);
 }
 else if (ilAutomatica && (now - lastPir > 120000)&& releState) {
     digitalWrite(rele,LOW);
     Serial.println("L - Se apagan las luces");
 }
}
```