Arduino en entorno IoT

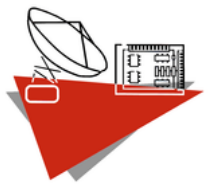
ACADEMIA BÁSICA DEL AIRE

Profesor:

Curso 2020-21

Objetivos

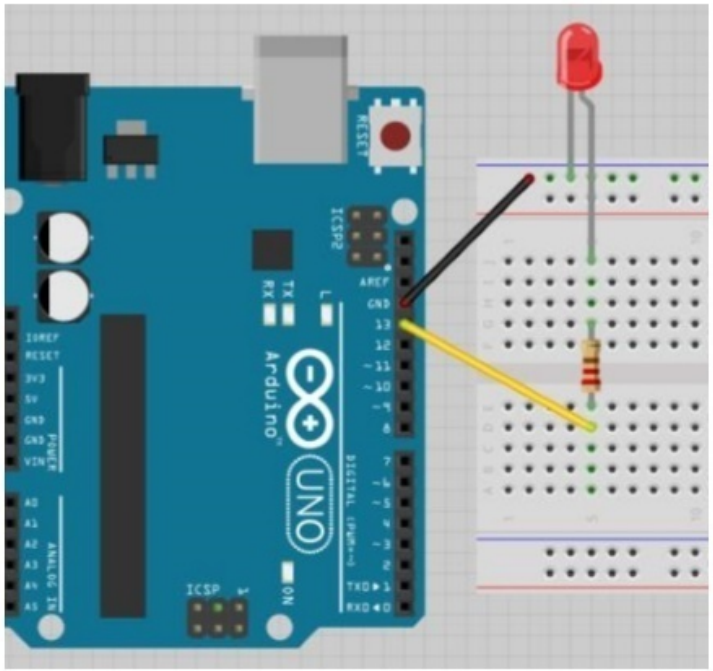
- Conocer los principios de los entornos IoT, “el Internet de las cosas”.
- La comunicación Ethernet con dispositivos Arduino.
- Conocer los sensores y actuadores básicos y su interacción con el entorno.



EJERCICIO 1. Led intermitente

Se trata de realizar un ejercicio básico que consiste en encender y apagar un LED de manera intermitente en intervalos de un segundo, conectándolo al PIN nº 3.

Material: Led y resistencia 220 ohm.



EJERCICIO 1

Las entradas: Son sensores (o transductores) electrónicos o mecánicos que toman las señales (en forma de temperatura, presión, humedad, contacto, luz, movimiento, pH etc.) del mundo físico y las convierten en señales de corriente o voltaje.
Por ejemplo un sensor de temperatura, un pulsador, un potenciómetro, un sensor de movimiento entre muchos más.

Las salidas: Son actuadores u otros dispositivos (también transductores) que convierten las señales de corriente o voltaje en señales físicamente útiles como movimiento, luz, sonido, rotación entre otros.
Por ejemplo: un motor que gire, un led o sistema de luces que se encienda automáticamente cuando esté oscureciendo, un buzzer que genere diversos tonos.

EJERCICIO 1

- **Declaración de variables**

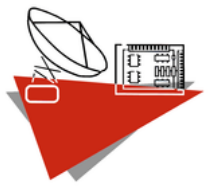
Todas las variables tienen que declararse antes de que puedan ser utilizadas. Declarar una variable es definir su tipo como int(entera), long (largo), flota (coma flotante), etc, asignándoles siempre un nombre, y, opcionalmente, un valor inicial

```
int led1=3; //LED en el pin digital 3
```

- **Función pinMode ()**

Asignación de pines digitales: Esta instrucción es utilizada en la parte de configuración setup() y sirve para configurar el modo de trabajo de un PIN pudiendo ser INPUT(entrada) o OUTPUT(salida).

```
pinMode(led1,OUTPUT); //  configura pin 3 como salida
```



EJERCICIO 1

- **Función digitalWrite ()**

Envía al ´pin´ definido previamente como OUTPUT el valor HIGH o LOW (poniendo en 1 o 0 la salida). El pin se puede especificar ya sea como una variable o como una constante (0-13).

digitalWrite(led1,HIGH);// deposita en el 'pin' un valor HIGH (alto o 1)

- **Retardo delay ()**

Detiene la ejecución del programa la cantidad de tiempo en ms que se indica en la propia instrucción. De tal manera que 1000 equivale a 1seg.

delay(1000); // espera 1 segundo

SOLUCIÓN 1

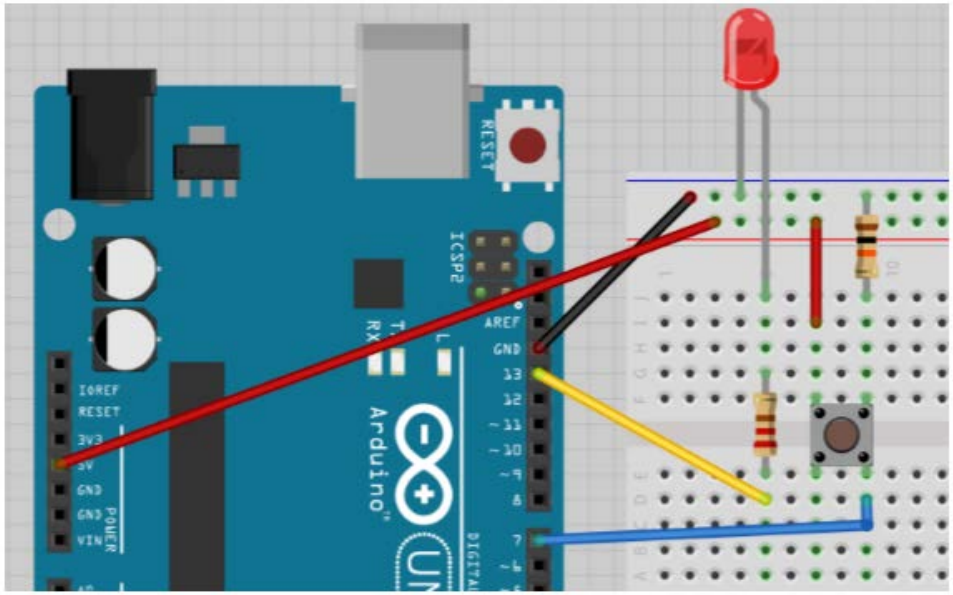
```
//ZONA GLOBAL
int led1=3; //  LED en el pin digital  3
//FUNCIÓN PRINCIPAL
void setup()    // Se ejecuta cada vez que el Arduino se inicia
{
pinMode(led1,OUTPUT); // Inicializa el pin 3 como una salida
  }
//FUNCIÓN CÍCLICA
void loop()    // Esta función se mantiene ejecutando cuando este energizado el Arduino
{
digitalWrite(led1,HIGH);  // Enciende el LED
delay(1000);            // Temporiza un segundo (1s = 1000ms)
digitalWrite(led1,LOW);   // Apaga el LED
delay(1000);            // Temporiza un segundo (1s = 1000ms)
 }
 // Fin del programa
```

EJERCICIO 2. Activación de led con pulsador

Se pretende controlar el encendido de un LED mediante un pulsador.

Material: Led y resistencia 220 ohm.

Pulsador y resistencia de 10K.



EJERCICIO 2

- **Declaración de variables**

Todas las variables tienen que declararse antes de que puedan ser utilizadas. Declarar una variable es definir su tipo como int (entera), long (largo), flota (coma flotante), etc., asignándoles siempre un nombre, y, opcionalmente, un valor inicial.

```
int led1=3;
int pulsador=5;
```

- **Función pinMode ()**

Asignación de pines digitales: Esta instrucción es utilizada en la parte de configuración setup() y sirve para configurar el modo de trabajo de un PIN pudiendo ser INPUT(entrada) o OUTPUT(salida).

```
pinMode(led1,OUTPUT); // declara  LED  como salida
pinMode(pulsador,INPUT); //  declara  pulsador como entrada
```

EJERCICIO 2

- **Función digitalWrite(pin, value)**

Envía al ´pin´ definido previamente como OUTPUT el valor HIGH o LOW (poniendo en 1 o 0 la salida). El pin se puede especificar ya sea como una variable o como una constante (0-13).

```
digitalWrite(led1,  HIGH);//  enciende el LED
digitalWrite(led1,  LOW); //  apaga el LED
```

- **Función digitalRead(pin)**

Lee el valor de un pin (definido como digital) dando un resultado HIGH (alto) o LOW (bajo). El pin se puede especificar ya sea como una variable o una constante (0-13).

```
digitalRead(pulsador)  ==  HIGH;//  testea si la entrada esta activa HIGH
```

EJERCICIO 2

- **Retardo delay ()**

Detiene la ejecución del programa la cantidad de tiempo en ms que se indica en la propia instrucción. De tal manera que 1000 equivale a 1seg.

delay(1000); // espera 1 segundo

- **if(si)**

If es un estamento que se utiliza para probar si una determinada condición se ha alcanzado, como por ejemplo averiguar si un valor analógico está por encima de un cierto número, y ejecutar una serie de declaraciones (operaciones) que se escriben dentro de llaves, si es verdad. Si es falso (la condición no se cumple) el programa salta y no ejecuta las operaciones que están dentro de las llaves.

El formato para if es el siguiente:

if(una variable toma un valor)

{

Ejecuta Instrucciones;

}

SOLUCIÓN 2

```
int led1=3;//  pin 3 asignado para el LED de salida
int pulsador=5;//  pin 5 asignado para el pulsador
void setup() // Configura entradas y salidas
{
 pinMode(led1,OUTPUT); //  declara  LED  como salida
pinMode(pulsador,INPUT); //  declara  pulsador como entrada
 }
void loop()
 {

for (int i=0; i<=5; i++)
 if(digitalRead(pulsador) == HIGH) //  testea si la entrada esta activa HIGH
     {
                digitalWrite(led1,HIGH);//  enciende el LED
                delay(1000);//  espera 1 segundo
                digitalWrite(led1,LOW); //  apaga el LED
      }
}
```

EJERCICIO 3. Lectura serial

Visualizar por la pantalla del ordenador a través del monitor serial un mensaje en espacios de 1 segundo.

EJERCICIO 3

- **Serial.begin(rate)**

Abre el puerto serie y asigna la tasa de baudios para la transmisión de datos serie. La típica tasa de baudios para comunicarse con el ordenador es 9600 aunque otras velocidades están soportadas.

```
void setup()
{
Serial.begin(9600); //abre el puerto serie y ajusta la tasa de datos a 9600 bps
 }
```

Nota: Cuando se usa la comunicación serie, los pines digitales 0 (Rx) y 1 (Tx) no pueden ser usados al mismo tiempo.

- **Serial.println(data)**

Imprime datos al puerto serie, seguido de un retorno de carro y avance de línea automáticos. Este comando toma la misma forma que Serial.print(), pero es más fácil para leer datos en el Serial Monitor.

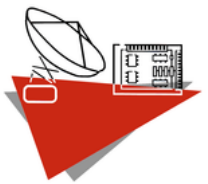
```
Serial.println("     "); //envía el mensaje que escribimos entre comillas
```

EJERCICIO 3

- **Retardo delay ()**

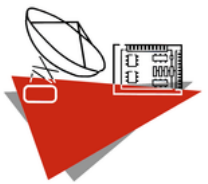
  Detiene la ejecución del programa la cantidad de tiempo en ms que se indica en la propia instrucción. De tal manera que 1000 equivale a 1seg.

  delay(1000); // espera 1 segundo



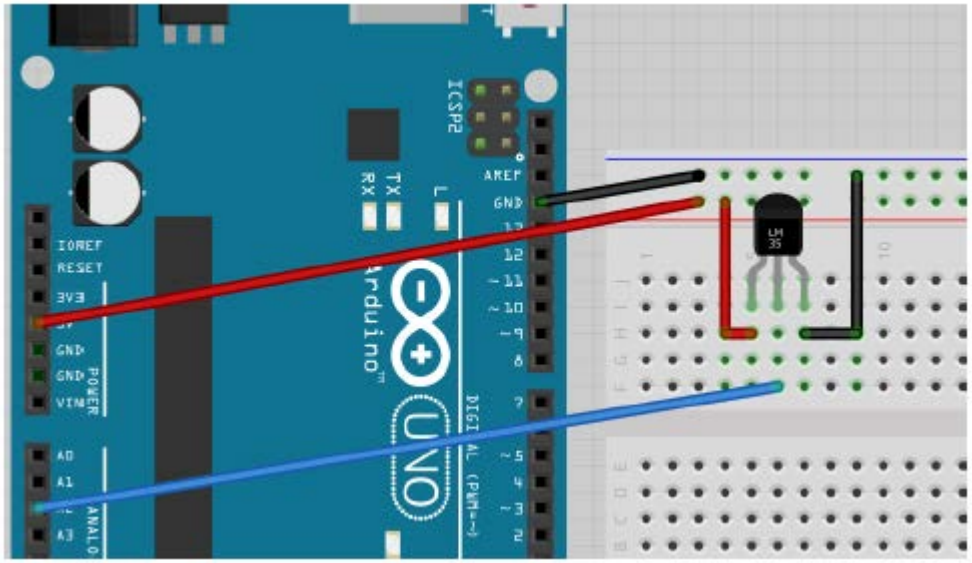
SOLUCIÓN 3

```
void setup()
{
Serial.begin(9600);// configura el puerto serie  a   9600bps
 }
void loop()
 {
Serial.println("Mucha suerte COMPIS“); //  imprime en el monitor serial "Mucha suerte 30" con un retorno de carro y linea nueva.
delay(1000);//  espera 1 segundo
}
```



EJERCICIO 4. Sensor Temperatura

Un sensor de temperatura es una resistencia que varia su valor en función de la temperatura exterior.
Tenemos que pasar el valor de la lectura (0 a 1023) a milivoltios (0 a 5000) lo cual nos hace la función matemática map(). Dividiendo dicho valor entre 100 nos da el valor de la lectura en grados.



EJERCICIO 4

```
int led1 =3;
float sensorTemp = A2;    // seleccionamos la entrada analógica 0
```

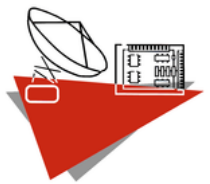
Como la temperatura la expresamos con decimales, utilizaremos un tipo nuevo de variable

```
int valorLectura;
float temperatura;
float milivoltios;
float temperatura_referencia = 24.00;
void setup()
{
   Serial.begin(9600);
  pinMode(led1,OUTPUT);
 }
```

EJERCICIO 4

```
void loop()
{
   valorLectura = analogRead(sensorTemp); // leemos el valor del potenciometro
  Serial.print("valor de lectura : ");
  Serial.print("\t "); //añadimos una tabulación
  Serial.println(valorLectura);
  milivoltios = map (valorLectura, 0, 1023, 0, 3300);
  temperatura = milivoltios / 100;
  Serial.print ("temperatura: ");    //Imprimimos en el puerto serial
  Serial.print ("\t"); // tabulación
  Serial.print (temperatura);                              //los resultados
  Serial.println(" grados");
  delay(1000);

 if (temperatura< temperatura_referencia)
        {
   digitalWrite(led1, HIGH);
        }
 else
        {
     digitalWrite(led1, LOW);
       }
```



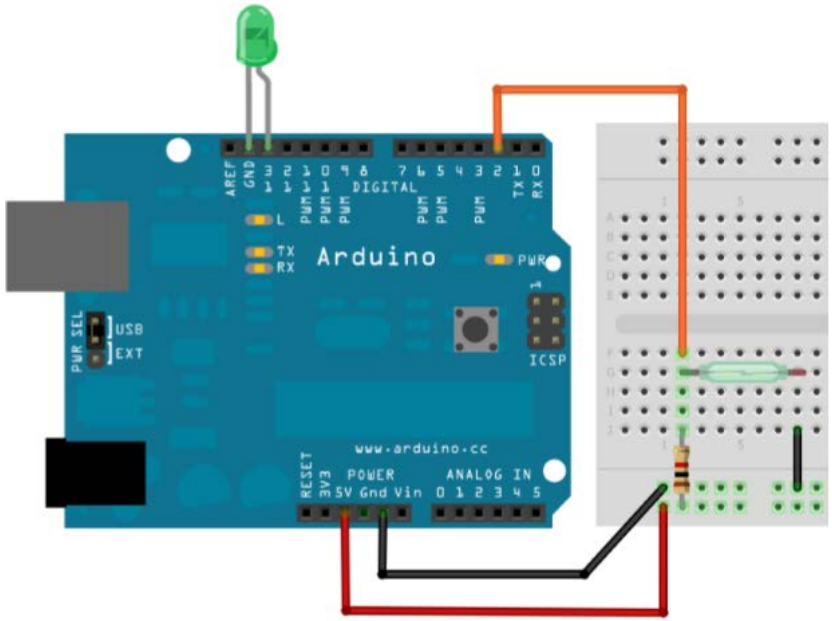
EJERCICIO5. Detector magnético

Reed switch es un interruptor eléctrico activado por un campo magnético, por ejemplo con un imán. Cuando los contactos están normalmente abiertos se cierran en la presencia de un campo magnético; cuando están normalmente cerrados se abren en presencia de un campo magnético.

Un uso muy extendido se puede encontrar en los sensores de las puertas y ventanas de las alarmas antirrobo, el imán va unido a la puerta y el reed switch al marco. La alarma se debe disparar cuando un campo magnético altere el estado del reed switch.

Detector magnético

Generar una alarma visual con un LED a partir de un campo magnético generado a un reed switch. La alarma se debe disparar cuando un campo magnético altere el estado del reed switch.



Detector magnético

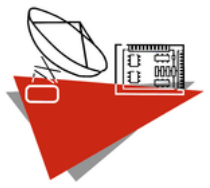
```
int contacto = 2;  //Pin asignado al reed switch
int led= 13;      //Pin asignado al LED
void setup()
 {
 pinMode(contacto,INPUT); //El reed switch como una entrada
 pinMode(led, OUTPUT);    //El LED como una salida
 }
void loop()
{
 if (digitalRead(contacto)==LOW) // Si el imán se acerca al reed switch
      {
for(int a=0; a<50; a++)//Ciclo for que va de 0 a 50, se repite mientras a sea menor a 50
{                  digitalWrite(led,HIGH); //se enciende el LED
delay(50);
          digitalWrite(led,LOW); //se apaga el LED
delay(50); //Tiempo
}
      }
else // Si el imán esta lejos del reed switch
      {         digitalWrite(led,LOW); //Mantiene apagado el LED
       }
}
```

EJERCICIO6. W5100

El W5100 es un controlador de Ethernet fabricado por Wiznet que podemos emplear con un procesador como Arduino para implementar comunicación por internet.

El W5100 está diseñado para facilitar la implementación de conectividad a internet sin necesidad de un SO, lo que lo hace interesante para MCU y aplicaciones de IoT.

El W5100 es un chip ampliamente empleado en dispositivos conectados en aplicaciones industriales, domésticas de domótica e IoT. Por ejemplo, entre otros muchos, pantallas y televisores inteligentes, relés activados por internet, impresoras, cámaras IP o dispositivos de almacenamiento en red.



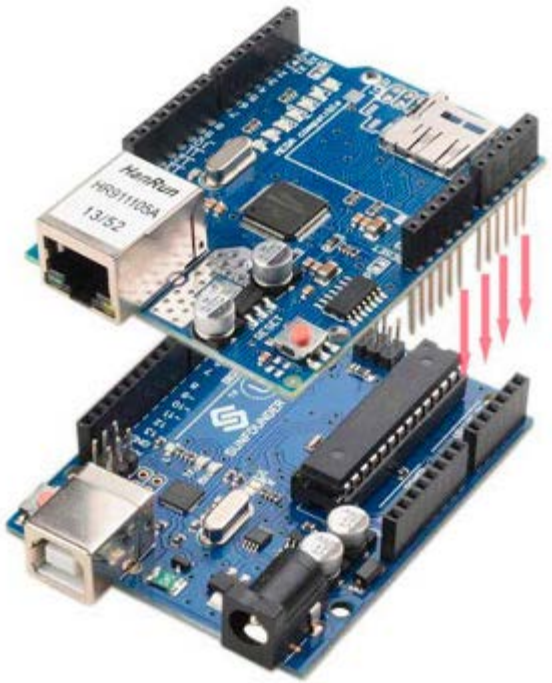
W5100

Incluye una pila de TCP/IP por hardware y buffer interno de 16Kbytes para Tx/Rx. Algunos módulos incorporan un lector de tarjeta SD, donde podemos guardar ficheros (html, txt, jpg, png) con los que trabajar cuando actuemos como servidor.

Admite velocidades de 10/100 Mbits/s, soportando modos Full-Duplex y Half-Duplex con detección y corrección automática de la polaridad. Cumple con las especificaciones IEEE 802.3 10BASE-T y 802.3u 100BASE-TX. La pila TCP/IP soporta conexiones TCP, UDP, IPv4, ICMP, ARP, IGMP and PPPoE, y hasta 4 conexiones simultáneas.

W5100

El más habitual es el Ethernet Shield, que tiene la ventaja de ser muy sencillo de conectar ya que es apilable. Normalmente estos módulos incluyen lector de micro SD. Como desventaja, sólo nos servirán si empleamos un Arduino UNO o Mega



W5100

SERVIDOR ETHERNET – CONTROLAR SALIDAS

El este ejemplo Arduino actúa también como servidor, pero esta vez queremos que el usuario pueda realizar acciones sobre Arduino a través de la página web que servimos.
En este caso, vamos a controlar una salida digital, a la que podemos conectar un Led para visualizar la respuesta.
Para ello, en primer lugar servimos la página web de forma similar al ejemplo anterior, pero en esta incluimos dos botones para cada salida.
Al pulsar en cada botón se realiza una nueva solicitud a Arduino, con diferente URL a la original. Arduino captura la nueva solicitud, y emplea la URL recibida para realizar las acciones oportunas.

W5100

SERVIDOR ETHERNET – CONTROLAR SALIDAS

El este ejemplo Arduino actúa también como servidor, pero esta vez queremos que el usuario pueda realizar acciones sobre Arduino a través de la página web que servimos.

En este caso, vamos a controlar dos salidas digitales, a las que podemos conectar un Led para visualizar la respuesta.

Para ello, en primer lugar servimos la página web de forma similar al ejemplo anterior, pero en esta incluimos dos botones para cada salida. Al pulsar en cada botón se realiza una nueva solicitud a Arduino, con diferente URL a la original. Arduino captura la nueva solicitud, y emplea la URL recibida para realizar las acciones oportunas.