Nodos de ejemplo: Centinela para plantas

Este nodo presenta un sistema que permite monitorizar el nivel de humedad en el sustrato de una planta.

Podemos encontrar el código en github. https://github.com/turbolargoo/Free-Connect/tree/master/Ejemplos%20proporcionados

Para este ejemplo hay disponibles tres archivos diferentes.

1. El primero y más básico nos permite ver el nivel de humedad desde la interfaz.
2. El segundo nos permite ver el nivel de humedad y además envía eventos si se supera un valor para alertarnos.
3. El tercero permite ver el nivel de humedad, envía eventos a la interfaz y mensajes por Telegram directamente a los usuarios.

Se presuponen conocimientos básicos sobre programación de Arduino.

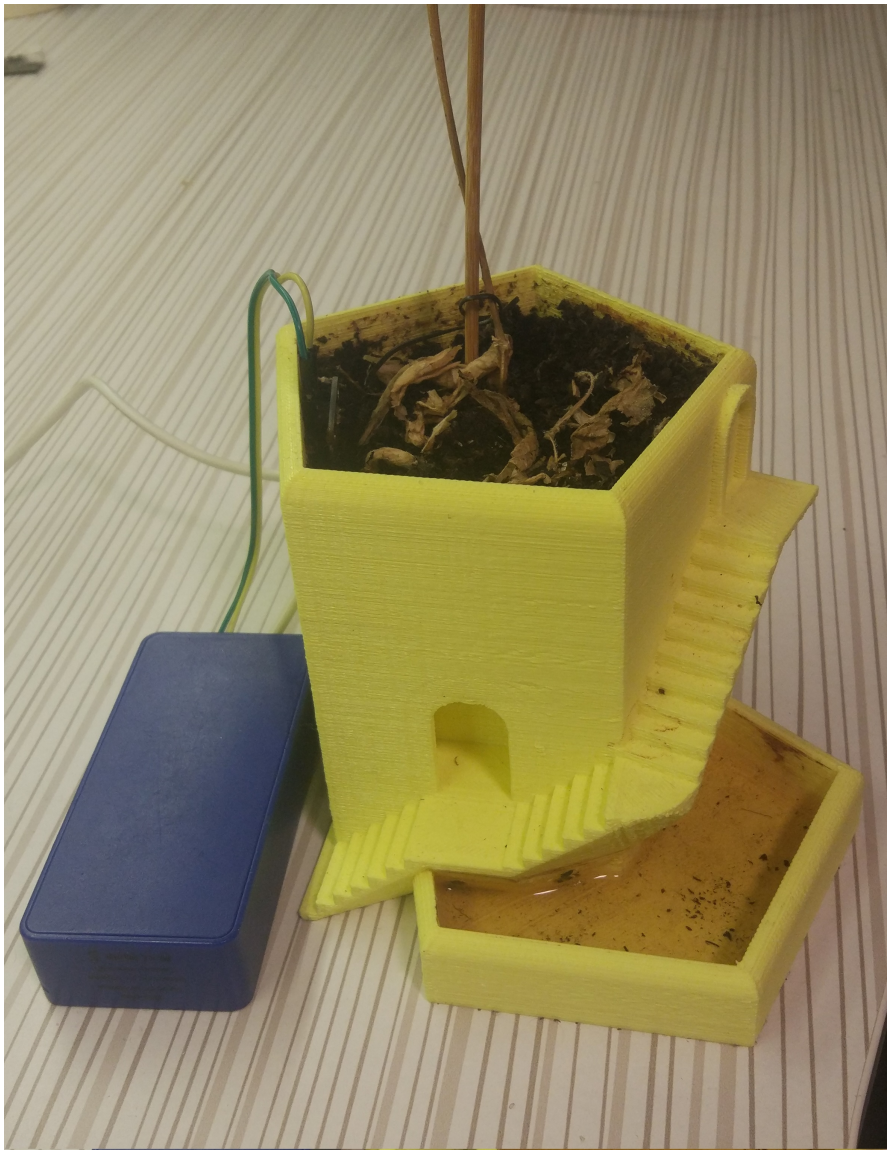

Figura 1: Nodo centinela para plantas

Características

- Lectura de humedad absoluta.
- Tiempo entre medidas programable.
- Diferentes mecanismos de alerta de nivel bajo.

Hardware empleado

- Placa esp32.
- Transistor PNP.
- Sensor de humedad.

Librerías adicionales

- QuickSortLib. Pantalla oled.

  Descargamos de `https://github.com/luisllamasbinaburo/Arduino-QuickSort`

Consideraciones sobre el sensor de humedad

Como ya comentamos en el apartado dedicado a estos sensores, los que basan su funcionamiento en electrodos descubiertos (ver figura), sufren grandes problemas de corrosión. Sin entrar en demasiados detalles, la corriente que circula por ellos acelera el proceso de corrosión, llegando a dejar el sensor inservible tras sólo un par de meses.

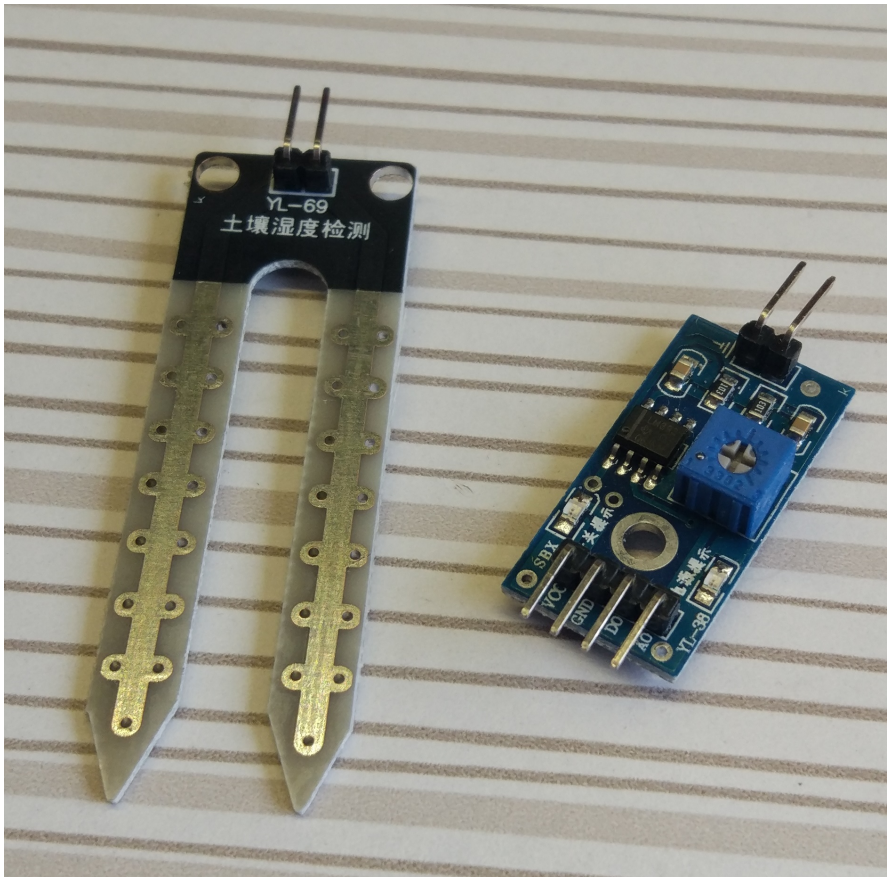

Figura 2: Sensor de humedad de electrodos descubiertos

Para mitigar este efecto, usaremos un transistor para poder cortar el paso de corriente por el sensor cuando no vayamos a hacer una medición. Con este método conseguimos alargar la vida útil considerablemente (aproximadamente un año).

El efecto de corrosión podría incluso afectar a nuestra planta, dependiendo de los metales utilizados en su construcción. Por este motivo no es recomendable su uso en plantas dedicadas al consumo humano.

Considerando estos datos, recomendamos utilizar un sensor del tipo capacitivo aunque para el ejemplo se haya utilizado uno con electrodos.

En el caso de usar uno capacitivo la única modificación necesaria será eliminar el transistor usado para controlar la alimentación del sensor ya que lo conectaremos directamente.

Aunque no tienen electrodos al descubierto, los componentes electrónicos, soldaduras y bordes de la placa sí están expuestos al ambiente alta humedad al

que vamos a someterlos. Es aconsejable proteger estos aspectos por ejemplo, con esmalte de uñas transparente. Creará una capa protectora duradera y podremos quitarlo fácilmente con acetona si fuera necesario (ver figura).

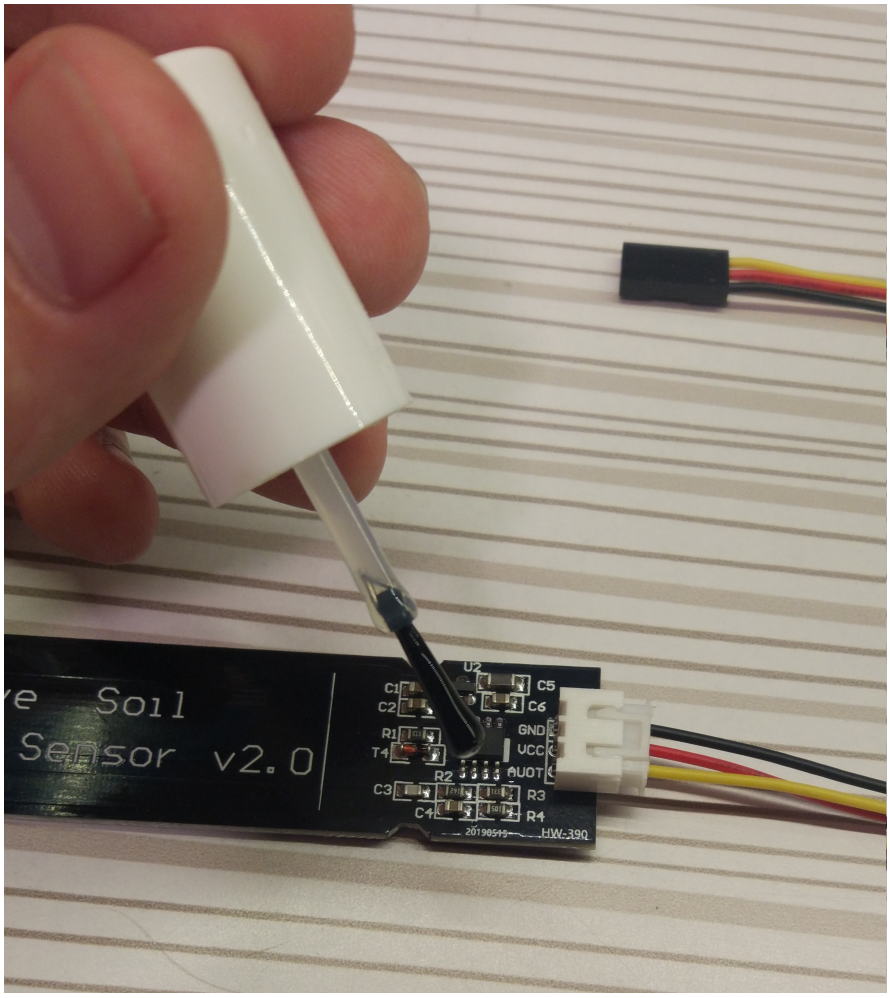

Figura 3: Protección aplicada a sensor capacitivo

Circuito eléctrico

A continuación planteamos los dos circuitos posibles.

- **Circuito para sensor de electrodos descubiertos:** Como ya hemos descrito, vamos a utilizar un transistor para poder encender o apagar el sensor.

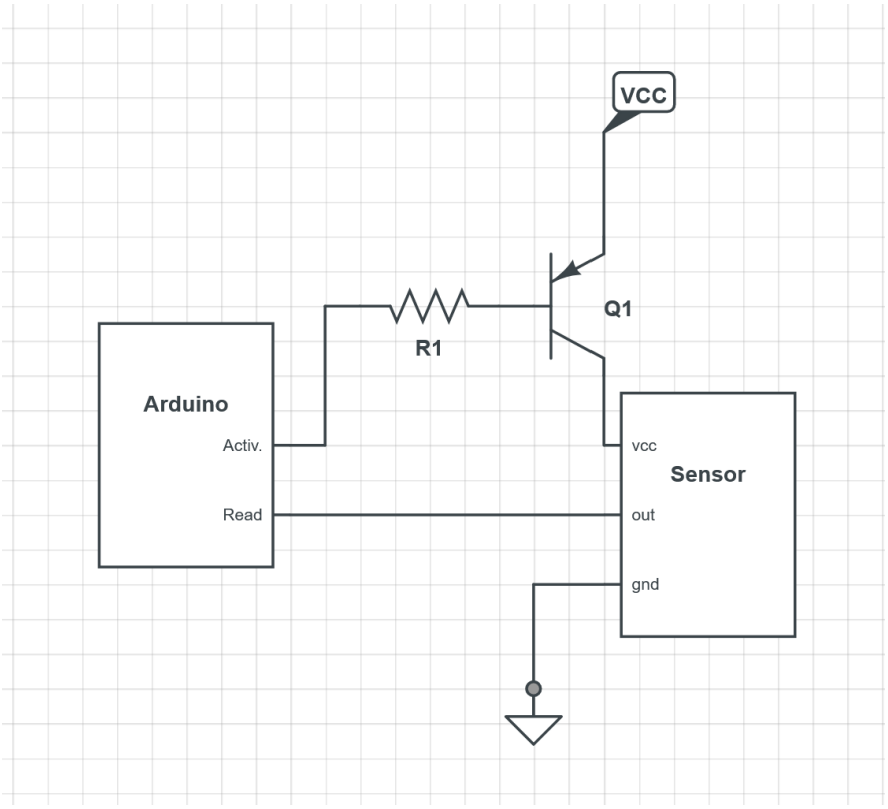

Figura 4: Circuito para sensor de electrodos descubiertos

- **Circuito para sensor capacitivo:**

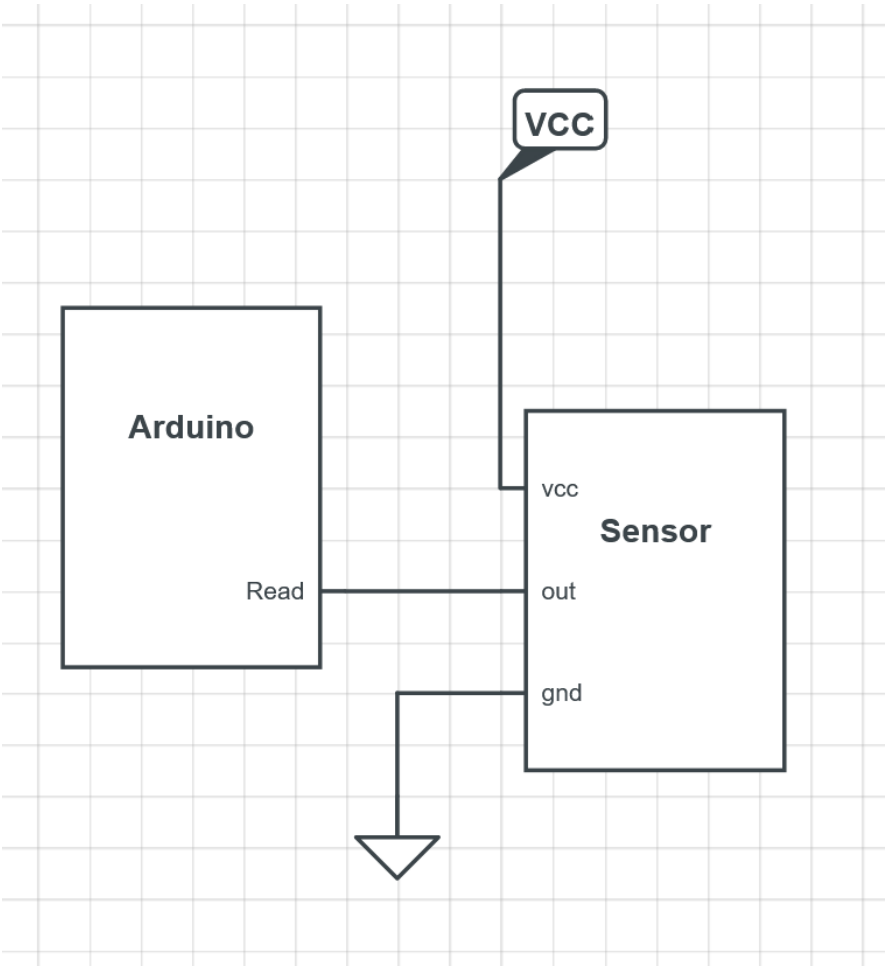

Figura 5: Circuito para sensor capacitivo

Programación

Partimos del código base para este desarrollo. A continuación se muestra el código añadido.

Archivo de configuración

Añadimos una serie de líneas:

1. **Configuración de pines:** Ubicación de el transistor de activación y de la lectura del sensor.

   const int humSensor = 33;

   const int activateSensor = 18;

2. **Variables utilizadas en el código.**

   - **boolean notified = false;**Esta variable se utiliza para notificar una sola vez la necesidad de regar la planta.
   - **int limWater=80** Nivel de humedad que se debe alcanzar para notificar de la necesidad de riego. Este valor dependerá entre plantas, tipo de sustrato... Debemos encontrar el valor que se ajusta a nuestro caso mediante ensayo y error.
   - **const int autoComp=3600000;** Tiempo que debe pasar entre comprobaciones.
   - **long tCheck=-autoComp;** Instante de tiempo cuando se hizo la última comprobación. Se inicializa con el valor inverso del tiempo entre comprobaciones para que en la primera iteración del bucle loop tras el arranque del nodo, cuando el tiempo es cercano a 0, se efectúe una medición.

3. **Parámetros de configuración webthings:**

   - `const char deviceName[] ="girasol";` Definimos el nombre que tendrá el nodo.
   - **WebThingAdapter *adapter;**
   - `const char *capacidades[] = {"MultiLevelSensor", nullptr};` Agregamos la capacidad de "Sensor Multinivel".
   - `ThingDevice Sensor("humedad1", "girasol", capacidades);`
   - `ThingProperty Humidity("Humidity", "Lectura del sensor", NUMBER, "LevelProperty");` Creamos la propiedad "Humidity"donde guardaremos el valor.
   - `ThingEvent NeedWater("Not enough moisture","Need to water" ,BOOLEAN, "AlarmEvent");` En el ejemplo que utiliza un evento para alertar al usuario, añadimos esta instrucción.

4. **Configuración de telegram** Introducimos los parámetros necesarios: `#define bOTtoken "token"` uTLGBot Bot(bOTtoken); const int debugLevelBot = 0; const int numChats=2; //we can specify to send the messages to more than one account //we specify here the chat ids we want the messages sent to `const char chatID[]={"chat"};` `const char chatID2[]={"chat"};` `const char *chatIDs[numChats]={chatID,chatID2};`

Setup

Añadimos las líneas de código para inicializar los pines convenientemente.

En la sección de configuración webthings, añadimos la siguiente línea:

Humidity.unit= "percent"; Cambia las unidades del valor mostrado en la pantalla a porcentaje

Loop

En este tipo de proyectos es muy importante realizar el control de tiempos guardando el resultado de la función *millis()* y comparándolo con los valores predefinidos en la configuración en lugar de utilizar delays. Esto es basar las condiciones de ejecución en el control de tiempo del microprocesador en vez de pararlo.

A la vista de la siguiente figura:

```
ArduinoOTA.handle();
if (millis()> tCheck + autoComp){
  if(water()){//reads the sensor and checks if the level is below threshold
    if(!notified){
      if(verboseOn)Serial.println("Watering needed");
      ThingPropertyValue value;
      value.boolean = true;
      ThingEventObject *ev = new ThingEventObject("NeedWater", BOOLEAN, value);
      Sensor.queueEventObject(ev);
      for (int i=0; i<numChats; i++) {
        Bot.sendMessage(chatIDs[i], "Water the plant");
      }
    }
    notified=true;
  }else{
    notified=false;
  }
  tCheck=millis();
}
adapter->update();
```

Figura 6: Código en la sección loop del ejemplo Centinela para plantas

Observamos que la primera línea después de la llamada a actualización OTA, comprueba si ha pasado el tiempo suficiente para realizar una nueva lectura.

El recuadro verde contiene el código que envía el evento.

El recuadro rojo, contiene el envío de la alerta por telegram.

Funciones

En este archivo se encuentran las funciones llamadas desde el loop. Comentaremos los aspectos mas relevantes sobre las funciones:

- **readSensor:** Para obtener una lectura lo más precisa posible realizamos múltiples lecturas y tras descartar las más altas y las más bajas, promediamos las restantes.

Primero encendemos el sensor mediante el transistor y seguido tomamos el número de medidas especificadas.

Utilizamos la librería quickSortLib para ordenar el vector de valores obtenido.

Recorremos el vector para calcular la media empezando en el elemento número 10 y acabando en el elemento n-10.

Utilizamos la función map para convertir el rango de valores de las mediciones (0-4096) en un porcentaje (rango 0-100).

Actualizamos el valor en la interfaz.

```
int readSensor(){
  int  readings[times];
  digitalWrite(activateSensor,HIGH);
  for(int i=0;i<times;i++){
    readings[i]=analogRead(humSensor);
    delay(10);
  }
  digitalWrite(activateSensor,LOW);
  QuickSort<int>::SortAscending(readings, 0, times-1);
  int avg=0;
  int count=0;
  for(int i =10;i<times-10;i++){//discard the 10 highest and lowest readings and average the rest
    avg+=readings[i];
    count++;
  }
  ThingPropertyValue value;
  int calculatedValue=map(avg/count, 0, 4096, 0, 100);
  value.number = calculatedValue;
  Humidity.setValue(value);
  if(verboseOn)Serial.println(calculatedValue);
  return calculatedValue;
}
```

Figura 7: Función para lectura de sensor de humedad

- **Water** Compara el límite establecido de humedad y con la lectura calculada del sensor con la siguiente instrucción: `return readSensor()>limWater;`

Coste de este nodo

Descripción	Unidades	Coste unitario €	Coste total €
Placa esp32	1	9	9
Placa perfboard	1	2	2
Transistor	1	0,25	0,25
Resistencias	1	0,1	0,1
Sensor Humedad	1	0,9	0,9
Impresión 3D	1	2	2

Tabla 1: Coste del nodo Centinela para plantas

Gastos totales de material: 14,25€

Nota: En caso de utilizar el sensor capacitivo (8€) el coste sería de 22€

Propuestas para mejora

En este ejemplo se proporciona la funcionalidad básica para este nodo pero hay muchas maneras de añadir características como por ejemplo:

El porcentaje definido como límite para el aviso de regar se podría modificar desde la interfaz para poder ajustarlo más cómodamente.

Añadir un mecanismo para habilitar/deshabilitar las alertas.