HORT-SOST II

Renovación dispositivo lluvia-viento

ACTUALIZACIÓN

Diseño y puesta a punto de dispositivo de medición de parámetros atmosféricos para implantación en huerta de la UMA Ciencias.

Dani Rodríguez Carrión
Departamento Arquitectura de Computadores

Contenido

Resumen

En esta primera actividad se propone renovar el dispositivo ya existente en el huerto encargado de tomar medidas relativas al viento y cantidad de agua caída por precipitaciones. Hablamos de una estación meteorológica compuesta por tres sensores:

- ➔ Una veleta (V) usada para medir la dirección y el ángulo de incidencia del viento.
- ➔ Un anemómetro (A) de copa para el cálculo de la velocidad del viento.
- ➔ Un pluviómetro (P) para medir la cantidad de agua llovida.

Se parte de una situación previa comentada en el siguiente apartado del documento y se propone una solución hardware consistente en un filtro paso-bajo para la salida del pluviómetro y del anemómetro. Para la veleta, se sigue el esquemático propuesto por el propio fabricante.

En el punto de diseño software se comentará el filtrado por software usado, así como algún punto importante del propio código.

Antecedentes

Los sensores A y P funcionan de manera similar. Son interruptores reed que, al accionarse por una acción determinada, bien que haya viento o llueva, mandan un pulso al microcontrolador. Usando interrupciones por GPIO se pueden contar dichos pulsos y así promediar las lecturas y calcular los valores de velocidad de viento y cantidad de agua caída (mm) en la estación.

Partimos de una estación montada en la que la lectura del pluviómetro poseía cierta cantidad de ruido que a la entrada del microcontrolador leía como valor alto y contaba como un pulso. Esto inducía falsas lecturas, indicando que llovía cuando estaba despejado. Los valores de lluvia y viento son relevantes en esta estación pues, con ellos, calculamos el coeficiente de evapotranspiración. Este coeficiente es útil para saber cuanta agua necesitamos emplear para el riego.

El cambio ha resultado en una renovación del circuito hardware que se nos proporciona por el fabricante a uno propio, para mitigar los rebotes innecesarios, además de añadir alimentación directa a 12V.

Por otro lado, se ha implementado un filtro antirrebotes por software para mejorar la lectura, previamente filtrada.

Diseño hardware

Como nos indica el fabricante, tanto el P como el A se debe de conectar a una resistencia pull-up de 100K y un condensador de 10pF, con VCC a 3.3V. Este esquemático obedece a la necesidad de tener el GPIO a nivel HIGH en reposo y cuando el sensor conmute, pasará a un nivel LOW, contándose esta caída como una pulsación.

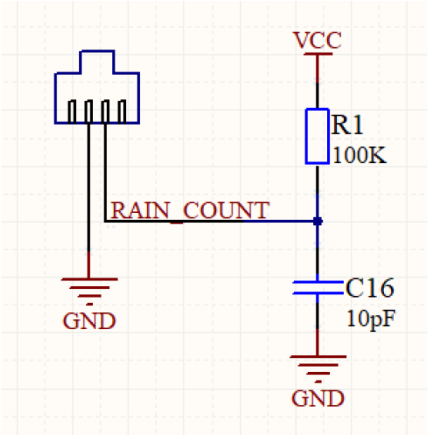

Ilustración 1. Esquemático según el fabricante

Sin embargo, en procesos posteriores de testeo, este circuito no es válido. El filtrado del condensador por si solo no es suficiente e induce mucho ruido. Por ello se plantea un filtro paso bajo a la entrada del GPIO, tal y como se muestra en la figura siguiente:

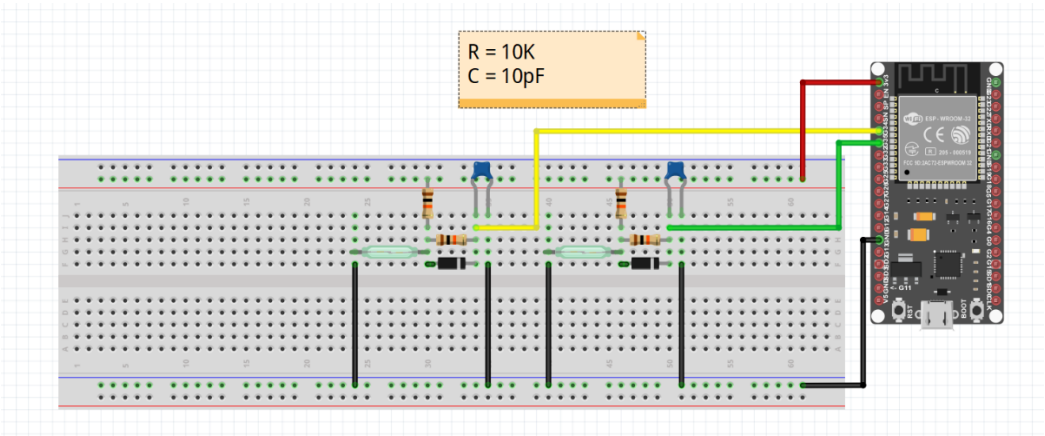

Ilustración 2. Esquemático del circuito antirrebotes

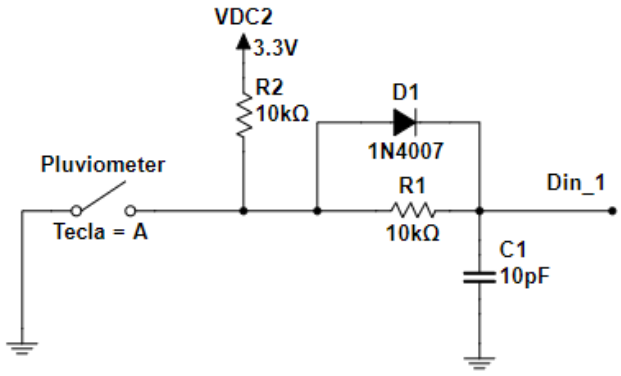

Ilustración 3. Circuito de la conexión de los sensores A y P

Donde se mantiene la resistencia pull-up con un valor de 10K, más que suficiente para asegurar un nivel HIGH a la entrada del GPIO. La configuración del filtro dio resultados óptimos para las lecturas de los sensores y la contabilización de conmutaciones.

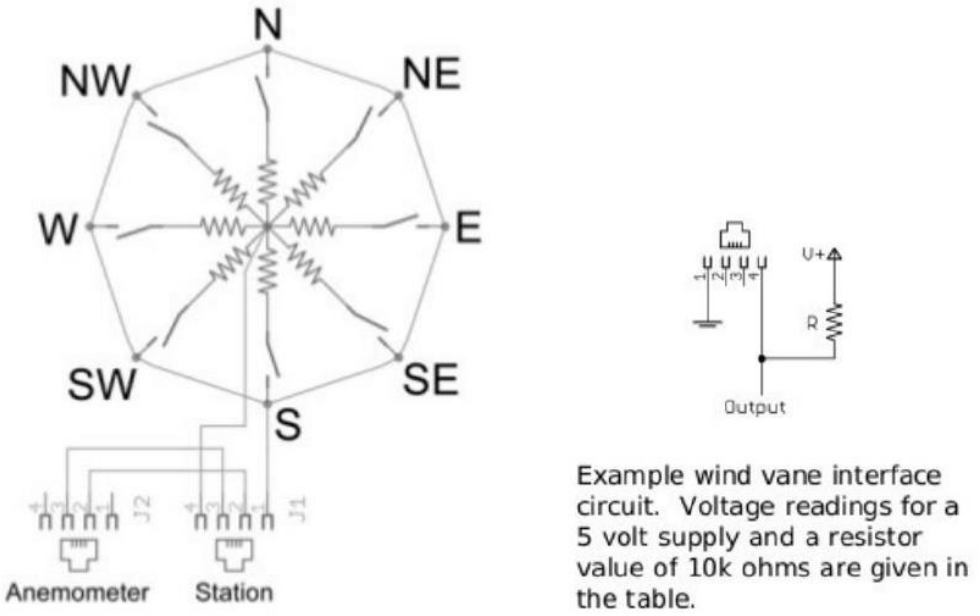

Ilustración 4. Configuración de la veleta.

Para la veleta, V+=3.3V, por lo que la tabla que nos da el fabricante debe de ser recalculada. Abajo un ejemplo de los valores leídos por el ADC del microcontrolador:

```
// ESP32 A/D values read for each wind vane direction
#define VANE_AD_N 2950
#define VANE_AD_NE 1660
#define VANE_AD_E 200
#define VANE_AD_SE 560
#define VANE_AD_S 965
#define VANE_AD_SW 2330
#define VANE_AD_W 3830
#define VANE_AD_NW 3460
```

GPIO	SENSORES
32	VELETA
34	PLUVIÓMETRO
35	ANEMÓMETRO

Diseño software

Como se comentó, el filtro antirrebotes por software es el que se muestra a continuación. Después de que ocurra el primer flanco esperado, esperamos el tiempo necesario para que el siguiente rebote se estabilice antes de comenzar a recibir nuevas interrupciones. También vale la pena señalar que estas variables deben definirse como volátiles para indicarle al compilador que no optimice la ejecución y, por lo tanto, evite valores inexactos durante las interrupciones de hardware.

```
void countRainCycles() {
  if (nextTimeRainIterrupt == 0 || nextTimeRainIterrupt < millis()) {
    rainCyclesCounter++;
    nextTimeRainIterrupt = millis() + 100;
  }
}

void countAnemometerCycles() {
  if (nextTimeAnemometerInterrupt == 0 || nextTimeAnemometerInterrupt
< millis()) {
    anemometerCyclesCounter++;
    anemometerMinuteCyclesCounter++;
    nextTimeAnemometerInterrupt = millis() + 10;
  }
}
```

Conclusiones

Se ha conseguido solucionar el problema de lectura de los sensores de manera eficiente, con pocos componentes pasivos y un código sencillo. Además, se ha alimentado directamente con una fuente de 12V continua, proporcionándole mayor robustez energética.