```
/* Arduino sketch for Weather device from Sparkfun.
Uses only the wind direction vane and the anemometer (not the rain gauge).
Although the inclination for a weather logger is to run it for
a long time, due to the way Wiring.c implements the millis() function,
this should be restarted, oh, monthly. The millis() functions overflows
after about 49 days. We could allow for that here, and handle the
wraparound, but you've got bigger problems anyway with the delay()
function at an overflow, so it's best to "reboot".
=========================================================
ANEMOMETER
=========================================================
This is connected to Arduino ground on one side, and pin 2 (for the
attachInterrupt(0, ...) on the other.
Pin 2 is pulled up, and the reed switch on the anemometer will send
that to ground once per revolution, which will trigger the interrupt.
We count the number of revolutions in 5 seconds, and divide by 5.
One Hz (rev/sec) = 1.492 mph=0.44704.
=========================================================
WIND DIRECTION VANE
=========================================================
We use a classic voltage divider to measure the resistance in
the weather vane, which varies by direction.
Using a 10K resistor, our ADC reading will be:
   1023 * (R/(10000+R))
where R is the unknown resistance from the vane. We'll scale
the 1023 down to a 255 range, to match the datasheet docs.
                  +5V
                   |
                   <
                   >     10K
                   <   Resistor
                   <
                   >
                   |
 Analog Pin 5------|
                   |
                   -----------| To weather vane
                              | (mystery resistance)
                   -----------|
                   |
                   |
                 -----
                  ---
                   -
The ADC values we get for each direction (based on a 255 max)
follow, assuming that pointing away from the assembly center
is sector zero. The sector number is just which 45-degree sector
it is, clockwise from the "away" direction. The direction
shown is assuming that "away" is West. Depending how
you orient the system, you'll have to adjust the directions.
Sector   Reading  Direction
  0        18       W
  1        33       NW
  2        57       N
  7        97       SW
  3       139       NE
  6       183       S
  5       208       SE
  4       232       E
The values in the ADC table below list the midpoints between
these, so our reading can vary a bit. We'll pick the first value
that's >= our reading.
=========================================================
RAIN GAUGE
=========================================================
Not implemented here. Hey. I live in Seattle. It's ALWAYS raining. Who
cares how much?
Okay, it would probably be done the same way as the anemometer, and use
attachInterrupt(1, ...) on pin 3. Each interrupt represents
.011 inches of rain, according to the docs.
*********************************************************************/
#include <TimeLib.h>

#define uint  unsigned int
#define ulong unsigned long

#define PIN_ANEMOMETER  2     // Digital 2
#define PIN_RAINGUAGE  3     // Digital 2
#define PIN_VANE        A3     // Analog 5

// How often we want to calculate wind speed or direction
#define MSECS_CALC_WIND_SPEED 5000
#define MSECS_CALC_WIND_DIR   5000

volatile int numRevsAnemometer = 0; // Incremented in the interrupt
ulong nextCalcSpeed;                // When we next calc the wind speed
ulong nextCalcDir;                  // When we next calc the direction
ulong time;                         // Millis() at each start of loop().
ulong hourTime = 3600000;

float numBuckettip = 0;
float rainAmount = 0.0;
float rainRate = 0.0;

// ADC readings:
#define NUMDIRS 8
ulong   adc[NUMDIRS] = {26, 45, 77, 118, 161, 196, 220, 256};

// These directions match 1-for-1 with the values in adc, but
// will have to be adjusted as noted above. Modify 'dirOffset'
// to which direction is 'away' (it's West here).
char *strVals[NUMDIRS] = {"W","NW","N","SW","NE","S","SE","E"};
byte dirOffset=0;

//=======================================================
// Initialize
//=======================================================
void setup() {
   Serial.begin(9600);
   pinMode(PIN_ANEMOMETER, INPUT);
   digitalWrite(PIN_ANEMOMETER, HIGH);
   pinMode(PIN_RAINGUAGE, INPUT);
   digitalWrite(PIN_RAINGUAGE, HIGH);
   attachInterrupt(0, countAnemometer, FALLING);
   attachInterrupt(1, countRainmeter, FALLING);
   nextCalcSpeed = millis() + MSECS_CALC_WIND_SPEED;
   nextCalcDir   = millis() + MSECS_CALC_WIND_DIR;
}

//=======================================================
// Main loop.
//=======================================================
void loop() {
   time = millis();

   if (time >= nextCalcSpeed) {
      calcWindSpeed();
      nextCalcSpeed = time + MSECS_CALC_WIND_SPEED;
   }
   if (time >= nextCalcDir) {
      calcWindDir();
      nextCalcDir = time + MSECS_CALC_WIND_DIR;
   }
   if(time >=hourTime){

   }
}

//=========================================================
// Interrupt handler for anemometer. Called each time the reed
// switch triggers (one revolution).
//=========================================================
void countAnemometer() {
   numRevsAnemometer++;
}
//=========================================================
// Interrupt handler for rainmeter. Called each time the reed
// switch triggers (one tip of bucket).
//=========================================================
void countRainmeter() {
static unsigned long lastmillis = 0;
unsigned long m = millis();
  if (m - lastmillis < 200){
// ignore interrupt: probably a bounce problem
  }else{
   numBuckettip++;
   rainAmount = numBuckettip*0.011;
   Serial.print("RAIN: ");
  Serial.println( rainAmount,3);
  }
  lastmillis = m;
}


//=========================================================
// Find vane direction.
//=========================================================
void calcWindDir() {
   int val;
   byte x, reading;

   val = analogRead(PIN_VANE);
   val >>=2;                        // Shift to 255 range
   reading = val;

   // Look the reading up in directions table. Find the first value
   // that's >= to what we got.
   for (x=0; x<NUMDIRS; x++) {
      if (adc[x] >= reading)
         break;
   }
   //Serial.println(reading, DEC);
   x = (x + dirOffset) % 8;   // Adjust for orientation
   Serial.print("  Dir: ");
   Serial.println(strVals[x]);
   // digital clock display of the time
  // setTime(hr,min,sec,day,month,yr); Another way to set
  /* setTime(9,15,0,29,5,2017); // Another way to set
 Serial.print(hour());
 printDigits(minute());
 printDigits(second());
 Serial.print(" ");
 Serial.print(day());
 Serial.print(" ");
 Serial.print(month());
 Serial.print(" ");
 Serial.print(year()); */
 Serial.println();
}


//=========================================================
// Calculate the wind speed, and display it (or log it, whatever).
// 1 rev/sec = 1.492 mph
//=========================================================
void calcWindSpeed() {
   int x, iSpeed;
   // This will produce mph * 10
   // (didn't calc right when done as one statement)
   long speed = 14920;
   speed *= numRevsAnemometer;
   speed /= MSECS_CALC_WIND_SPEED;
   iSpeed = speed;         // Need this for formatting below

   Serial.print("Wind speed: ");
   x = iSpeed / 10;
   Serial.print(x);
   Serial.print('.');
   x = iSpeed % 10;
   Serial.print(x);
Serial.print( " mph=");
Serial.print(x*0.4470);
Serial.print( " m/s");



   numRevsAnemometer = 0;        // Reset counter
}

void printDigits(int digits) {
  // utility function for digital clock display: prints preceding colon and
leading 0
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}
```