


Integrated electronic device/system for monitoring rainfall, humidity and/or drought

Kacper Bogusz Juśkiewicz

Nicolaus Copernicus University in Toruń, ULO, Szosa Chełmińska street 83, 87-100 Toruń, Poland
✉ kacpersky.ck@gmail.com,  <https://orcid.org/0000-0003-4742-0224>

Key words: high-resolution humidity measurements, fertilization, irrigation, soil reclamation, Arduino, Raspberry Pi, humidity measuring probe, monitoring rainfall and drought.

Introduction

Current climate changes, including drought, represent one of the most severe and extreme natural phenomena, directly affecting the environment, economy and society, and increasingly leading to soil degradation. In drought-stricken areas, losses in agricultural production are driving up food prices. The world's demand for water will continue to grow further in the coming decades. In view of the presented facts, there is a tremendous need for a systemic approach to saving water in the process of soil reclamation. The problem of fertilization is linked to reasonable irrigation. These processes ought to be synchronized and run simultaneously, under the control by the system. This paper presents a prototype measurement probe as an element of a measurement network.

The material complements the article "High-resolution humidity measurements in rational fertilization and irrigation methods as part of soil reclamation under climate change".

The project was based on the idea of DIY (do it yourself). This term is associated not only with the independent manufacture of devices - without the participation of professionals and huge financial resources, but also with a non-commercial and free approach - for everyone who would like to use the presented solution. The proprietary prototype of a network of devices for measuring the amount of water and monitoring drought has been designed in a way that guarantees: maximum simplicity, minimum costs, with a short time of execution, maximum repeatability. Each sensor collects nearly 2,000 results in one day. The designed probe, as of today, is equipped with 9 sensors that work continuously, around the clock, sending results every 45 seconds - this gives over 6,300,000 measurements per year for one device working in the network.

Probe components

The enclosure specification includes DN 150 PVC pipe, DN 32 PVC pipe, DN 110 PVC ventilator, DN 110 PVC sleeve, DN 110/75 PVC reduction, DN 110 PVC pipe and a glued universal plug closing it at the bottom (Fig. 1). The main element of the probe is the measuring part. It consists of the Arduino Uno Rev3 microcomputer with the programming environment (Arduino IDE) and a number of sensors, including a rain time and intensity measurement sensor (YL-83 with author's modifications) and the amount of rain (MarQuant 014369), temperature and humidity sensors air (DHT22), soil moisture sensors (HW-103 with author's modifications) at five different depths. Details of electrical connections and the software are presented in the supplementary (Table 1).

An important element of the structure is the housing, which must guarantee tightness, preventing water from entering the interior containing the measuring electronics. After many trials, pipes and fittings of water and sewage systems turned out to be the cheapest and, by definition, tight solution. The connections are tight as the elements contain gaskets. The measuring part of the device consists of the Arduino Uno Rev3 microcomputer with the programming environment (Arduino IDE) and a number of sensors that have been connected and programmed in accordance with the original design assumptions.

Proprietary modifications

The sensor that has been changed in a significant way is the sensor for measuring the time of occurrence and intensity of rain, in which the author used a completely proprietary design of the measuring detector. One of the simplest and most sensitive methods of rainfall detection was used, i.e. the resistance method. Its design is based on two systems of conductive paths, isolated from each other and placed at a short distance. This method is known. For example, double resistance combs are used, but after flooding with water, the detectors stop fulfilling their role and until they dry up, they incorrectly inform about precipitation. Therefore, a wire mesh with a horizontal and vertical resolution of 1 mm was used (Fig. 2). Such a wire spacing in the detector guarantees that even the smallest (2 mm) raindrops shorten the resistive mesh and splash on the mesh. The largest droplets, down to 5 mm, which have a larger surface area, also correctly indicate a greater intensity of precipitation. However, they close the circuit only during the actual rainfall, and after its completion, the drops gradually flow down, opening the circuit.

Another sensor modified by the author is the sensor for measuring soil moisture. In this case, a profile probe was made independently, from scratch, up to a depth of 110 cm, with point measurement for several depths: 10, 30, 50, 70 and 110 cm. The depths were determined by the lengths of the root systems of field crops. Determining the moisture content at these specific depths is extremely important as a well-hydrated plant makes better use of the nutrients from the soil.

Verification and correction of measurements

Before starting the actual measurements, several series of measurements were made to verify the accuracy of the results, which turned out to be consistent with the measurements of commercial devices and with the results from meteorological simulations and forecasts. The discrepancy between rainfall and humidity results was less than 1%, and air temperatures were less than 5% - despite the use of cheap electronic components. At the same time, the same results were obtained with other commercial devices. In order to calibrate the soil moisture sensors, soil moisture measurements were made using the weighing method. The weight of moist soil samples was determined after drying at a temperature of about 105 °C for 3 hours. In this way, a correction factor was obtained and, as a result, the actual result of soil moisture.

The above description allows you to recreate the probe and build the network yourself.

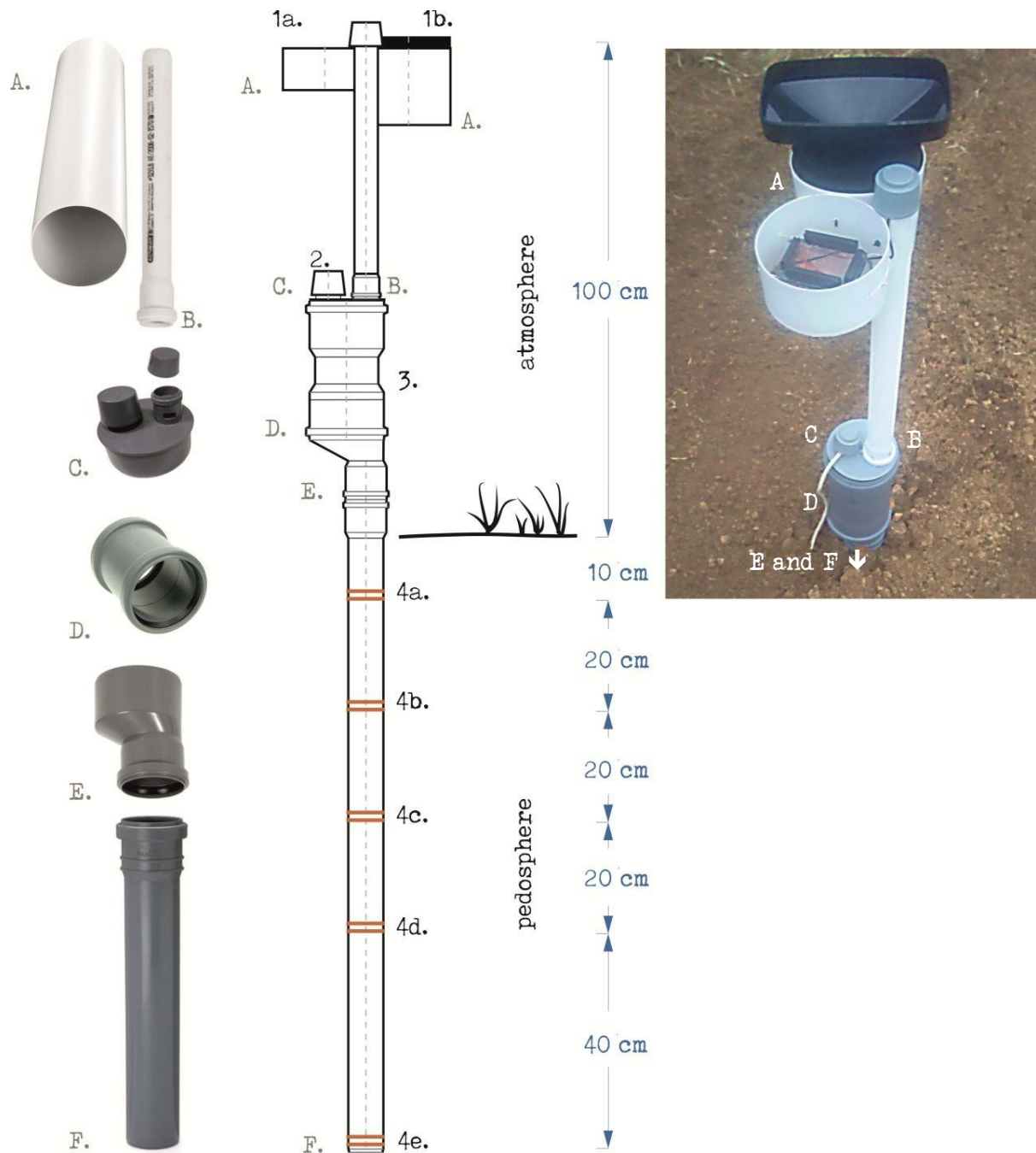


Fig. 1. Diagram of the proprietary casing, the basic element of the rainfall and drought monitoring system: sensors for measuring the duration and intensity of rain (1a) and the amount of rain (1b), sensors for measuring air temperature and humidity (2), data collection, processing and transmission system (3), soil moisture sensors at the depths of 10 cm (4a), 30 cm (4b), 50 cm (4c), 70 cm (4d), 110 cm (4e) and the materials used: PVC pipe DN 150 (A), PVC DN 32 (B), PVC ventilator DN 110 (C), PVC sleeve DN 110 (D), PVC reduction DN 110/75 (E), PVC pipe DN 110 closed with a cap (F). Own source

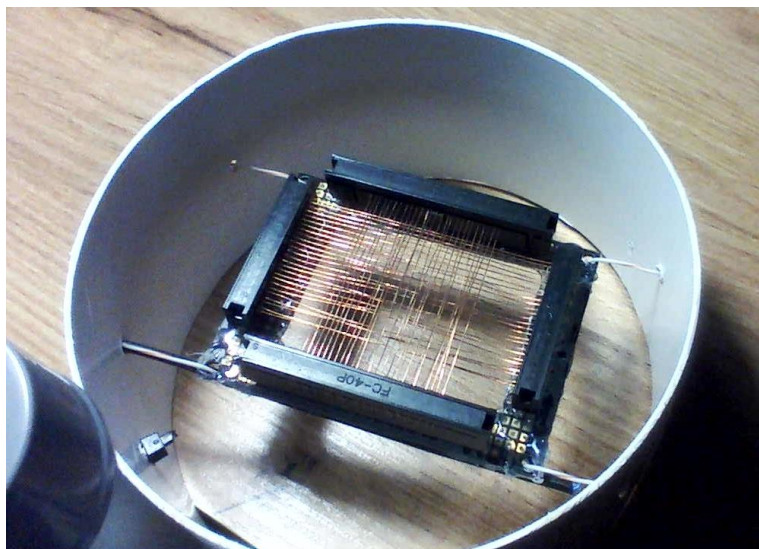
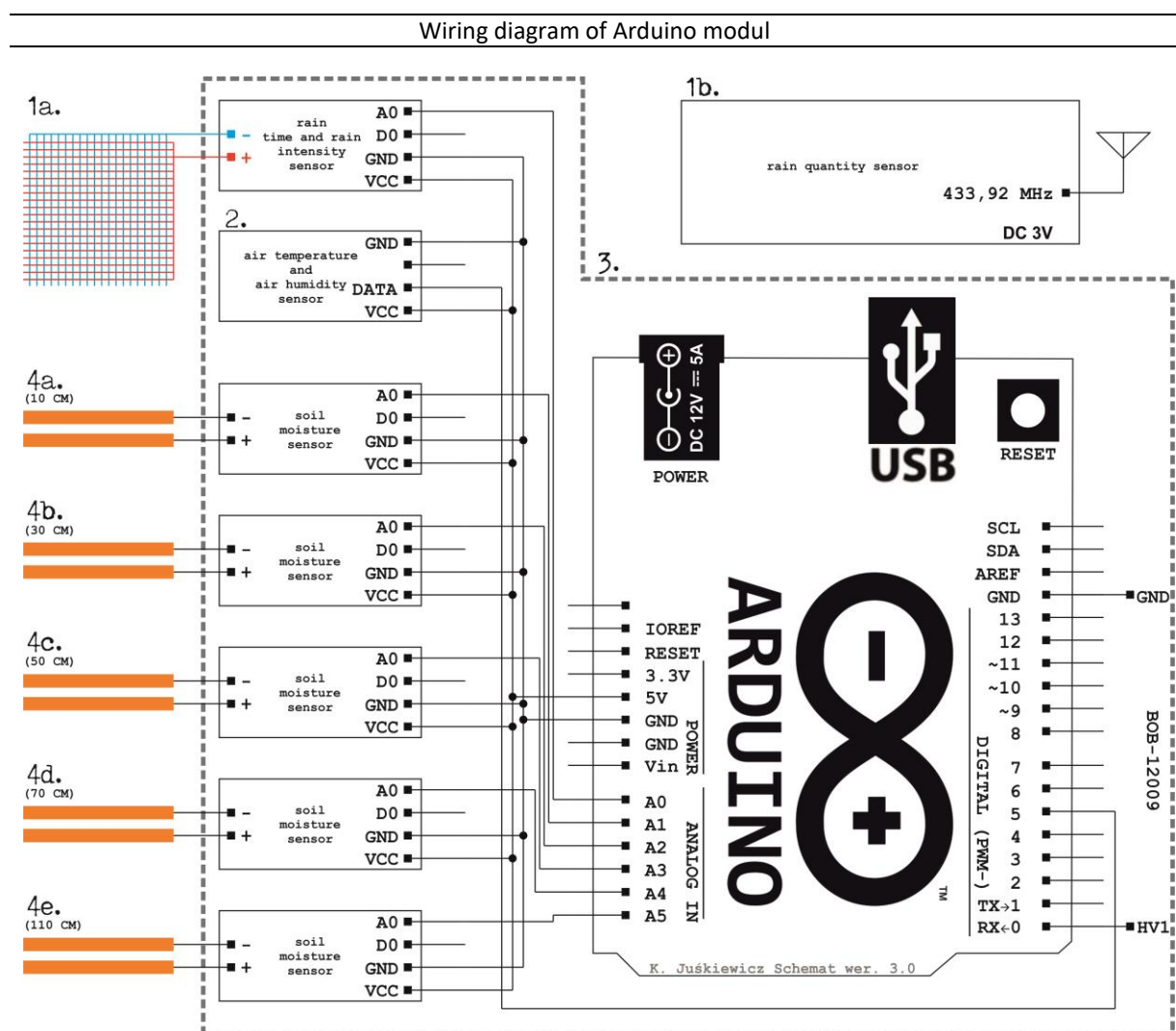


Fig. 2. Proprietary detector of the sensor for measuring the time of occurrence and intensity of rain. Own source

Table 1. Wiring diagrams and the source code of the soil moisture probe. Own source



Source code of Arduino modul

```
#include <dht.h>
dht DHT;

#define DHT22_PIN 5

float hum;
float temp;

int sensor_A0 = A0;
int sensor_D0 = 2;
int wartosc_A0;
int wartosc_D0;

int sensor_A1 = A1;
int sensor_D1 = 3;
int wartosc_A1;
int wartosc_D1;

int sensor_A2 = A2;
int sensor_D2 = 4;
int wartosc_A2;
int wartosc_D2;

int sensor_A3 = A3;
int sensor_D3 = 5;
int wartosc_A3;
int wartosc_D3;

int sensor_A4 = A4;
int sensor_D4 = 6;
int wartosc_A4;
int wartosc_D4;

int sensor_A5 = A5;
int sensor_D5 = 7;
int wartosc_A5;
int wartosc_D5;

int sensor_A6 = A6;
int sensor_D6 = 8;
int wartosc_A6;
int wartosc_D6;

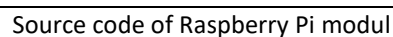
int sensor_A7 = A7;
int sensor_D7 = 9;
int wartosc_A7;
int wartosc_D7;

const int sensorMin = 0;
const int sensorMax = 1024;

void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);
  pinMode(8, INPUT);
}
```

```
void loop() {  
  
    int chk = DHT.read22(DHT22_PIN);  
    hum = DHT.humidity;  
    temp= DHT.temperature;  
    Serial.print("Wilgotność powietrza: ");  
    Serial.print(hum);  
    Serial.print(" %, Temperatura powietrza: ");  
    Serial.print(temp);  
    Serial.println(" st. Celsjusza");  
  
    wartosc_A0 = analogRead(sensor_A0);  
    wartosc_D0 = digitalRead(sensor_D0);  
  
    wartosc_A1 = analogRead(sensor_A1);  
    wartosc_D1 = digitalRead(sensor_D1);  
  
    wartosc_A2 = analogRead(sensor_A2);  
    wartosc_D2 = digitalRead(sensor_D2);  
  
    wartosc_A3 = analogRead(sensor_A3);  
    wartosc_D3 = digitalRead(sensor_D3);  
  
    wartosc_A4 = analogRead(sensor_A4);  
    wartosc_D4 = digitalRead(sensor_D4);  
  
    wartosc_A5 = analogRead(sensor_A5);  
    wartosc_D5 = digitalRead(sensor_D5);  
  
    wartosc_A6 = analogRead(sensor_A6);  
    wartosc_D6 = digitalRead(sensor_D6);  
  
    wartosc_A7 = analogRead(sensor_A7);  
    wartosc_D7 = digitalRead(sensor_D7);  
  
    int percent = map(wartosc_A0, sensorMax, sensorMin, 0, 100);  
    Serial.print(percent);  
    Serial.println("% intensywności opadu ");  
  
    int percent2 = map(wartosc_A1, sensorMax, sensorMin, 0, 100);  
    Serial.print(percent2);  
    Serial.println("% wilgotności gleby na gł. 10 cm ");  
  
    int percent3 = map(wartosc_A2, sensorMax, sensorMin, 0, 100);  
    Serial.print(percent3);  
    Serial.println("% wilgotności gleby na gł. 30 cm ");  
  
    int percent4 = map(wartosc_A3, sensorMax, sensorMin, 0, 100);  
    Serial.print(percent4);  
    Serial.println("% wilgotności gleby na gł. 50 cm ");  
  
    int percent5 = map(wartosc_A4, sensorMax, sensorMin, 0, 100);  
    Serial.print(percent5);  
    Serial.println("% wilgotności gleby na gł. 70 cm ");  
  
    int percent6 = map(wartosc_A5, sensorMax, sensorMin, 0, 100);  
    Serial.print(percent6);  
    Serial.println("% wilgotności gleby na gł. 110 cm ");  
  
    Serial.print("D0: ");  
    Serial.print(wartosc_D0);  
    Serial.print(" -- A0: ");  
    Serial.println(wartosc_A0);  
  
    Serial.print("D1: ");  
    Serial.print(wartosc_D1);  
    Serial.print(" -- A1: ");  
    Serial.println(wartosc_A1);  
}
```

Wiring diagram of Raspberry Pi modul



Citation: RepOD, <https://doi.org/10.18150/MF7ZGR>

```

import os
import fcntl
import struct
import RPi.GPIO as GPIO
import time
import signal
import sys
import serial

czas = time.time()
dane = [ ]
lfile1 = os.path.dirname(__file__) + '/log1.csv'
lfile2 = os.path.dirname(__file__) + '/log2.csv'
lfile2t = os.path.dirname(__file__) + '/log2.txt'

znacznik = 'CSV '

def zmiana(wejście):
    global czas
    global dane
    nowy_czas = time.time()
    dt = int(round((nowy_czas - czas) * 1000000))
    czas = nowy_czas
    dane.append(dt)
    i = 0
    for d in dane[-20:]:
        if d < 250:
            i += 1
    if i >= 10:
        nowe = dane
        dane = [ ]
        if len(nowe) >= 100:
            print(len(nowe), min(nowe[20:-21]), nowe)
            i = 0
            m = len(nowe) - 2
            w = ""
            d = [ ]
            while i < m:
                if nowe[i] >= 6400 or 9800 <= nowe[i] + nowe[i+1] <= 10200:
                    if len(w) >= 20:
                        ws = w
                        if len(w) >= 64:
                            ws += ' ' + format(int(w[:64], 2), '016x')
                        if
                            (int(w[0:8],2)+int(w[8:16],2)+int(w[16:24],2)+int(w[24:32],2)+int(w[32:40],2)+int(w[40:48],2)+int(w[48:56],2))%25
                            6 == int(w[56:64],2):
                                ws += ' OK'
                                ws += ' ' + format(int(w[:16], 2), '04x')
                                ws += ' ' + w[16:20]
                                ws += ' ' + str(int(w[20:23], 2))
                                ws += ' ' + w[23:24]
                                ws += ' ' + str(int(w[32:40] + w[24:32], 2))
                                t = ((int(w[48:56] + w[40:48], 2) - 900) / 10.0 - 32) * 5 / 9
                                ws += ' ' + str(t)
                                wl = str(int(w[32:40] + w[24:32], 2)) + ';' + str(t).replace('.', ',')
                                if wl != locals().get('wl_s', ""):
                                    wl_s = wl
                                    with open(lfile1, 'a') as f:
                                        f.write(time.strftime("%Y-%m-%d %T", time.localtime()) + ';' + wl + '\n')
                                print(ws)
                                w = ""
                                if nowe[i] >= 6400:
                                    i -= 1
                                elif 3300 <= nowe[i] + nowe[i+1] <= 3900:
                                    if nowe[i] >= nowe[i+1]:
                                        w += 'I'
                                else:

```



```
w += '0'
i+=2
print('-----')

def sigterm_handler(_signo, _stack_frame):
    sys.exit(0)

signal.signal(signal.SIGINT, sigterm_handler)
signal.signal(signal.SIGTERM, sigterm_handler)

try:
    GPIO.setmode(GPIO.BOARD)
    wejscie = 7
    GPIO.setup(wejscie, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.add_event_detect(wejscie, GPIO.BOTH, callback=zmiana)

    ser = serial.Serial('/dev/serial0', 9600)
    while True:
        line = ser.readline().decode('utf-8', errors = 'ignore').strip()
        print('<- ' + line)
        print('-----')
        with open(lfile2t, 'a') as f:
            f.write(time.strftime('%Y-%m-%d %T', time.localtime()) + '\t' + line + '\r\n')
        if line.startswith(znacznik):
            with open(lfile2, 'a') as f:
                f.write(time.strftime('%Y-%m-%d %T', time.localtime()) + ';' + line[len(znacznik):] + '\r\n')

except KeyboardInterrupt:
    pass

finally:
    GPIO.cleanup()
```