Design of an IoT system for monitoring chemical and physical properties of soil (moisture, temperature, pH), precipitation, pest intrusion for pineapple and tomato crops

Fidele Houeto
Jonathan Gheysens

May 2023

Contents

1 Introduction

In this report we will focus on how to design an Internet of Things-sytem to monitor the chemical and physical properties of soil (moisture, temperature and pH), precipitation and pest intrusion. We will focus on a design which can be used in pineapple and tomato plantations. First we will discuss all the types of required sensors. After this we will have a good overview of the sensors we can use in our own system. So, we provided a prototype of the IoT system we can use for our application and we will also analyse this system.

2 Types of sensors

2.1 Soil moisture sensor

You have two main types of soil moisture sensors. They differ from each other by using a different working principle.
The first type is a resistive soil moisture sensor. It uses the relationship between electrical resistance and water content to gauge the moisture levels. When there is a lot of water in the soil, this results in a higher electirical conductivity. So, when you send an electrical current from one probe to the other, a lower resistance reading is obtained.

A second type of soil moisture sensors is a capacitive sensor. A capacitive soil moisture sensor is commonly built with a positive and negative plate, which are separated by a dielectric medium in the middle. The soil humidity is a dielectric medium and its capacitance changes with moisture content. By measuring the change in charge and discharge time, we can determine the level of humidity of the earth is by reading an analog voltage with an Arduino board. [21]

In the following table you can find some examples of soil moisture sensors and a comparison of some of their properties.

Sensor	**Type**	**Operating Voltage**	**Current**	**Output Type**	**Price**
Grove Moisture Sensor[14]	Resistive	3.3-5V	35mA	Analog	$3.30
SEN-13322 (Sparkfun)[17]	Capacitive	3.3-5V	12mA	Analog	$6.50
VH400[6]	TDR-based[1]	3.5-20V	12mA	Analog	$41.95
YL-69[1]	Resistive	3.3V - 5V	N/A	Analog	$1.99
SEN0193[2] (DFRobot)	Capacitive	3.3-5V	5mA	Analog	$5.86

Table 1: Comparison of Soil Moisture Sensors

[1]For more information about how this sensor type works, see https://en.wikipedia.org/wiki/TDR_moisture_sensor

2.2 Soil temperature sensor

A possible way to measure the temperature of the soil is using a thermometer. Some of the thermometers normally used in soil work include mercury or liquid in glass, bimetallic, bourdon, and electrical-resistance thermometers. The selection of the appropriate thermometer for an application is based on its size, availability, accessibility to the measurement location, and the required degree of precision. For precise temperature measurements, thermocouples are preferred because of their quick response to sudden changes in temperature and ease of automation. But for our applications we can simply use a temperature sensor based on the working principle of a thermometer.[16] Most soil temperature sensors are combined with a soil moisture sensor.

Sensor	**Type**	**Operating Voltage**	**Current (max)**	**Output Type**	**Price**
MODBUS-RTU RS485 (S-Soil MT-02A)[3]	Capacitive	3.6-30V	24mA	Digital	$79.00
THERM200 (Vegetronix)[5]	Resistive	3-24V	3mA	Analog	$39.95
SMT01[13]	Capacitive	3.3-5V	150mA	Digital	$5.00

Table 2: Comparison of Soil Temperature Sensors

2.3 Soil PH sensor

Soil pH sensors measure the acidity or alkalinity of the soil by detecting the hydrogen ion activity. The ideal pH range for most plants falls between 5.5 and 7.5. [22] So before designing a full sensor system, we need to think about the necessity of a pH sensor, because the pH of the soil doesn't change quickly. That's why it might be better to just measure it by hand once a month for example and not implement it in an IoT-system.
In the following table some pH-sensors are compared to each other:

Sensor	**Type**	**Operating Voltage**	**Current**	**Output Type**	**Price**
RS-PH-N01-TR-1[4]	Potentiometric	5-30V	5-10mA	Digital	\$36.7
LSPH01[11]	Capacitive	3.3-5V	<10mA	Digital	\$165

Table 3: Comparison of Soil pH Sensors

2.4 Precipitation sensor

The most common sensors used for measuring rainfall are tipping bucket rain gauges and weighing precipitation gauges. Tipping bucket rain gauges consist of a funnel that collects rainwater and funnels it into a small seesaw-like device. Each time a set amount of water is collected, the device tips, emptying the water and recording the "tip" as a measurement of rainfall. Weighing precipitation gauges work by weighing the amount of precipitation that falls on a flat surface. As the weight of the precipitation accumulates, it is recorded and used to calculate the total amount of rainfall.[23]
Another type of sensor that can be used for measuring precipitation is the optical rain sensor. This is a sensor that uses infrared light to detect water hitting its surface. The infrared beams bounce within the sensor lens, and as water droplets hit the surface, the infrared light escapes through. The changes in the intensity of the infrared beams during rainfall are directly proportional to the size of the rain drop. This means that the system is capable of detecting very small rain drops.[23]
All the type of sensors mentioned before are quite expensive, but have a very high accuracy. They can measure how much rain sensor. If the measurements don't really need to be precise and you just want to know if it's raining or not, you can also use a simple rain sensor, such as an FC-37 rain sensor. The electronic board and collector board make up the two pieces of the sensor, with the collector board collecting the water drops. The resistance of the collector board varies according to the amount of water on its surface, providing accurate readings of rainfall. The sensor has both a digital output and an analog output, and comes equipped with a potentiometer that enables sensitivity adjustment of the digital output by adjusting the threshold value.[20]
Now, we will focus on some of the sensors who can be implemented in our system. In the following table you can find a comparison between some of these sensors:

Sensor	**Type**	**Operating Voltage**	**Current (max)**	**Output Type**	**Price**
Hydreon RG-11[8]	Optical	10-30V	$\leq$ 50 mA	Analog	$\sim$ \$59
Rain Sensor FC-37[20]	Resistive	3.3-5V	$\leq$ 15 mA	Analog	$\sim$ \$3
Tipping rain gauge (Sparkfun)[19]	Tipping bucket	5V	N/A	Analog	\$37.95

Table 4: Comparison of Rain Sensors

2.5 Pest control sensor

There are two types of device which can be used for pest control. The first type of devise is a motion detection sensor. This is an electrical device that utilizes a sensor to detect nearby motion. Such a device is often integrated as a component of a system that automatically performs a task or alerts a user of motion in an area. They form a vital component of security, automated lighting control, home control, energy efficiency, and other useful systems.
Motion sensors can be used as a means of crop pest control. Passive infrared (PIR) sensors are often used to detect the movements of wild animals in fields, such as deer, wild boar and rabbits, which can cause significant damage to crops. PIR sensors can trigger audible or visual alarms to scare animals away from crops.
In addition, vibration sensors can be used to detect the movements of insect pests that feed on crops. The sensors can be placed on plants to detect vibrations caused by insects and trigger automatic control measures, such as watering, spraying insecticides or activating traps.
Motion sensors can also be used in combination with other technologies to improve pest control. For example, vision sensors can be used to detect insects on plants and spray them with targeted insecticides, thereby reducing the amount of insecticide used and minimising adverse effects on the environment. In addition, the data collected by the sensors can be used to monitor pest activity and plan more effective control strategies in the future.
A second type of devise you can use for pest control is a pest repeller sensor. This device is designed to detect and repel pests such as rodents, insects, or other unwanted creatures. These sensors are typically built up of a motion detection sensor and then an extra pest repelling part. Pest repellers can be very useful,but their effectiveness may vary depending on the specific pests and the environmental conditions. Different pests may respond differently to various repelling technologies (e.g. ultrasonic waves, ground vibration...), and certain other factors like room layout, furniture, and noise interference can affect their performance. It's recommended to follow the manufacturer's instructions and consider other integrated pest management techniques for comprehensive pest control.

Table 5: Comparison of pest control sensors

Sensor	**Type**	**Operating Voltage**	**Current**	**Output Type**	**Price**
PIR Motion Sensor (SE062)[12]	Optical	5V	$\leq$ 5mA	Digital, Analog	\$11.29
HC-SR04[9]	Ultrasonic motion detector	3.3-5V	$\leq$ 65mA	Digital, Analog	$\sim$ \$5
SEN0171 (DFROBOT)[10]	Passive	3.3-5V	$\leq 15\mu A$	Digital	$\sim$ \$4.90
MA40S4S (MURATA)[18]	Ultrasonic transducer	5V	N/A	N/A	$\sim$ \$9

3 Deployment: concrete example

3.1 Design decisions

To read out all of our sensors, we also need a microcontroller board. In our design, we chose to use the Dramco-Uno board. This is a board based on the design of the Arduino Uno, but has some extra sensors on the board itself. For more information about the Dramco-Uno see https://dramco.be/projects/dramco-uno/.
To measure soil moisture and temperature we can use multiple sensors, but one of the sensors we've discussed before can measure both soil properties. The SMT-01 sensor can measure the temperature and by measuring the rate of dissipation of thermal energy which depends on the moisture content in the soil the sensor can compute the soil moisture level. The SMT-01 sensor is simple in design and consists of a heating component (bipolar transistor 2N2222A) and a thermometer (digital 1-Wire sensor DS18B20). This sensor only costs $5, so this is a perfect solution for our design.[13]

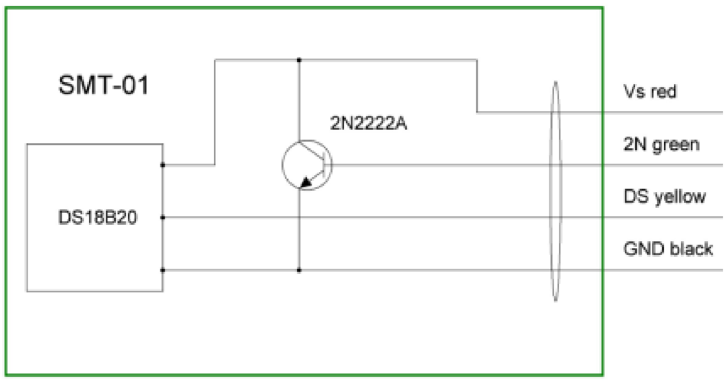

Figure 1: Internal structure of SMT-01 sensor

As third parameter we want to monitor the soil pH-level. In our design we won't use a pH-sensor, because these sensors are quite expensive. Secondly, this parameter doesn't change quickly, so we can simply measure this by hand e.g. every week. This solution reduces the cost of our design and has minor disadvantages.
To measure precipitation, we don't need a complicated sensor. It suffices to detect if it's raining or not, because we also measure soil moisture. Based on the data of soil moisture, we can estimate the precipitation. The FC-37 rain sensor is set up by two pieces: the electronic board and the collector board that collects the water drops. Basically, the resistance of the collector board varies accordingly to the amount of water on its surface. When it's raining, the

resistance increases, and so the output voltage will decrease. By measuring this analog value with our microcontroller board, we can see if it's raining or not by comparing this value with the treshold value.
To control the pest intrusion, we can start by using a motion detector sensor. The HC-SR04 ultrasonic sensor consists of an ultrasonic transmitter, a receiver and a control circuit. By sending a pulse on the trigger pin and then detecting the echo, we can determine how far an object is away from the sensor. With this sensor we will measure the distance and send this distance with LoRaWan to the cloud. We can further evaluate this data to see if there is pest intrusion or not.
When we detect pest intrusion, we can try to repel it by sending an ultrasonic wave. For this, we simply need an ultrasonic transducer and with the tone()-function of Arduino, we can create a signal for a certain frequency and duration on the correct pin.[7]

3.2 Design

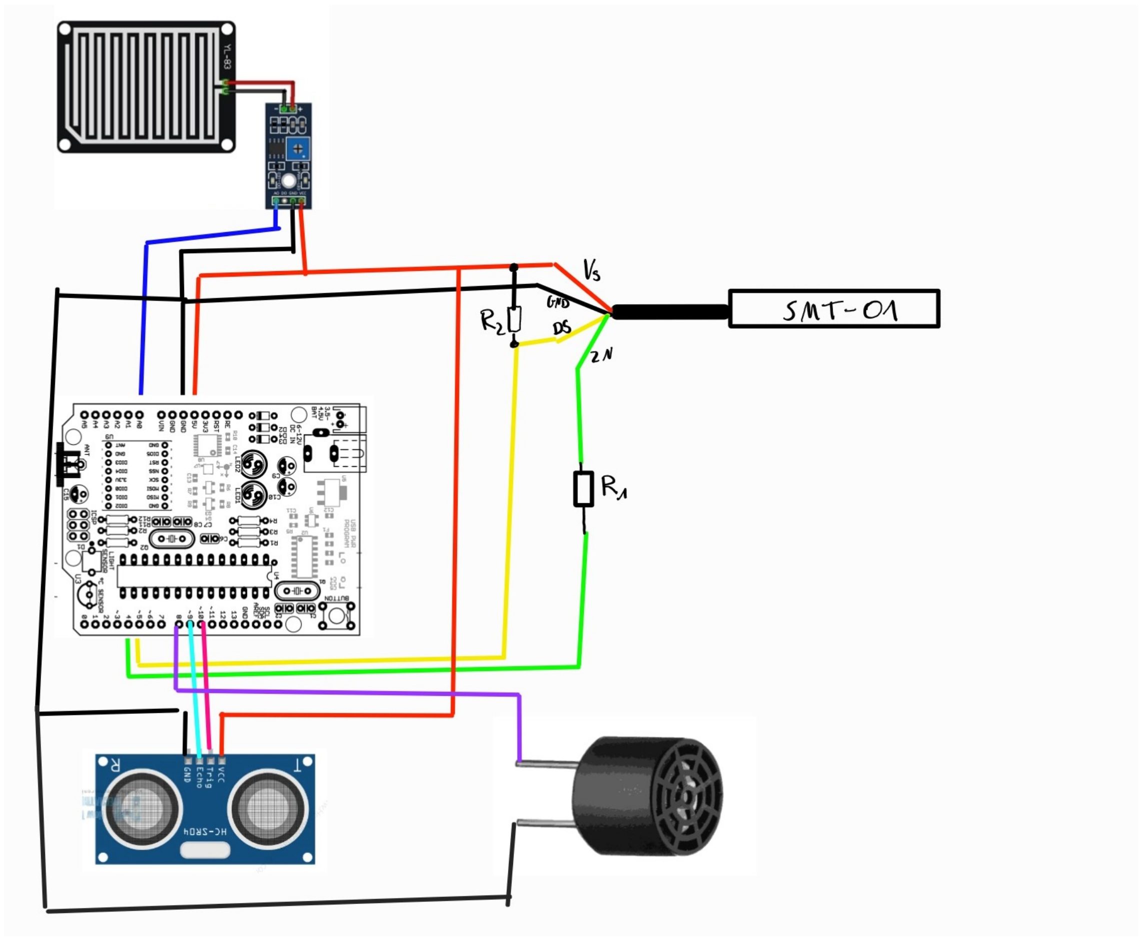

Figure 2: Full design

3.3 Code

3.3.1 Test Soil moisture & temperature sensor[13]

Listing 1: Test soil moisture & temperature sensor

```
/*
Soil Moisture & Temperature Sensor SMT-01
Developed by Oleksander Savinykh,
greensensorso@gmail.com

Example for Arduino UNO

SMT-01 use Heat Dissipation Method
```

```
Components of SMT-01:
DS18B20 - 1-Wire temperature sensor
2N2222A - as Heater

Connection wires of SMT-01 cable to Arduino UNO (see
    Electrical Circuit):
Yellow DS  - to Pin 5 (R2 4.7k - 2.0k to +5V, depending
    on length of the cable)
Green 2N   - to Pin 4 (via R1 (10k - 2.0k, depending on
    length of the cable)
Red        - to +5V
Black      - to GND

*/

#include <OneWire.h>

#define DARK  LOW
#define LIGHT HIGH
#define ON    HIGH
#define OFF   LOW

OneWire  ds(5);  // 1-Wire to Pin 5

byte i;
byte present = 0;
byte type_s;
byte data[12];
byte addr[8];
float celsius;
int m_err;

int pin_Led = 13;
int pin_Heater = 4;

int DS_found = 0;
float Time_Heat_Dissipation, Soil_Moisture,
    Soil_Temperature;
unsigned long Heating_Time = 30000; //ms
int j, k;

void DS18B20_init(void);
void DS18B20_measure(void);
void Measure_SMT (void);

unsigned long mtime;
unsigned long set_mtime;


void setup()
{

   pinMode(pin_Led,OUTPUT);
   pinMode(pin_Heater, OUTPUT);

   digitalWrite(pin_Led,DARK);
   digitalWrite(pin_Heater, LOW);

 // UART communication setup
   Serial.begin(9600);
   delay(10);
   Serial.println("");

   Serial.println("Initialization␣of␣DS18B20␣...␣");

   DS_found = 0;
   DS18B20_init();
   if (DS_found == 1){
       digitalWrite(pin_Led, LIGHT);
       Serial.println("Initialization␣is␣Ok");
       delay(1000);
       digitalWrite(pin_Led,DARK);
      }

   set_mtime = 1000;
   mtime = millis();

}//setup

void loop()
{

 if (millis() - mtime > set_mtime) {

  digitalWrite(pin_Led, LIGHT);

// Measurement

  if(DS_found == 1){

  Serial.println("Start␣measurement");
  Measure_SMT();

// Converting the Time of Heat Dissipation to Soil Moisture, %

// as example
  float Sensor_Dry = 250.0; //Time of Heat Dissipation for Dry Sensor
  float Sensor_Wet = 35.0;  //Time of Heat Dissipation for Wet Sensor

  Soil_Moisture = map(Time_Heat_Dissipation, Sensor_Dry, Sensor_Wet, 0.0, 100.0);
  if (Soil_Moisture < 0.0) Soil_Moisture = 0.0;
  if (Soil_Moisture > 100.0) Soil_Moisture = 100.0;

  Serial.print("Soil␣Moisture␣=␣");
  Serial.print(Soil_Moisture);
  Serial.println(",␣%");

  Serial.print("Temperature␣of␣Soil␣=␣");
  Serial.print(Soil_Temperature);
  Serial.println(",␣oC");
  }
  else{
   Serial.println("Next␣try␣to␣Initialization␣of␣DS18B20␣...␣");
   DS_found = 0;
   DS18B20_init();
   if (DS_found == 1){
       digitalWrite(pin_Led, LIGHT);
       Serial.println("Initialization␣is␣Ok");
       delay(1000);
       digitalWrite(pin_Led,DARK);
      }

    }

  set_mtime = 420000; // recommended pause between measurements, ms (7 minutes)
  mtime = millis();

  Serial.println("Waiting␣for␣the␣next␣measurement␣...");
  digitalWrite(pin_Led,DARK);

  }// if mtime

  delay(10);

}// loop


void DS18B20_init(void){ //
    ---------------------------------------------------------------

if ( !ds.search(addr)) {
    DS_found = 0;
    Serial.println("Sensor␣not␣found.");
    Serial.println();
    ds.reset_search();
    delay(250);
    return;
  }//if

Serial.print("ROM␣=");
  for( i = 0; i < 8; i++) {
    Serial.write('␣');
    Serial.print(addr[i], HEX);
  }

  if (OneWire::crc8(addr, 7) != addr[7]) {
      Serial.println("CRC␣is␣not␣valid!");
      DS_found = 0;
      return;
  }
  Serial.println();

// the first ROM byte indicates which chip
  switch (addr[0]) {
    case 0x10:
      Serial.println("␣␣Chip␣=␣DS18S20");  // or old
          DS1820
      type_s = 1;
      break;
    case 0x28:
      Serial.println("␣␣Chip␣=␣DS18B20");
      type_s = 0;
      DS_found = 1;
      break;
    default:
      Serial.println("Device␣is␣not␣a␣DS18x20␣family␣
          device.");
      return;
  }//switch

}//DS18B20_init

void DS18B20_measure(void) {//------------------------------------------------------

  m_err = 0;

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1);    // start conversion, with parasite power on at the end

  delay(1000);          // time need for conversion

  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);       // Read Scratchpad

  //Serial.print("  Data = ");
  //Serial.print(present, HEX);
  //Serial.print(" ");
  for ( i = 0; i < 12; i++) {           // we need 12 bytes resolution
    data[i] = ds.read();
    //Serial.print(data[i], HEX);
    //Serial.print(" ");
  }
  //Serial.print(" CRC=");
  //Serial.print(OneWire::crc8(data, 8), HEX);
  //Serial.println();


 if (OneWire::crc8(data, 8) != data[8]) {
      Serial.println("CRC␣is␣not␣valid!");
      m_err = 1;
  }


  // Convert the data to actual temperature
  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3; // 9 bit resolution default
    if (data[7] == 0x10) {
      // "count remain" gives full 12 bit resolution
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    // at lower res, the low bits are undefined, so let's zero them
    if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
    else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
    else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
    //// default is 12 bit resolution, 750 ms conversion time
  }

  celsius = (float)raw / 16.0;

  //Serial.print("  Temperature = ");
  //Serial.print(celsius);
  //Serial.print(" Celsius, ");
  //Serial.println();

 }//DS18B20_measure


void Measure_SMT ()
   {
    float t_current, tj;
    int m_cycle, j, m;
    unsigned long dtime;

    Time_Heat_Dissipation = 0.0;
    j = 0;
    m = 0;
    tj = 0.0;

    Serial.println("Temperature␣of␣Soil␣measurement␣...");
    for (m_cycle = 0; m_cycle<10; m_cycle++)
        {
          DS18B20_measure();
          if (m_err == 0)
             {
              tj = tj + celsius;
              j++;
              Serial.println(celsius);
             }//if
          else {m++;}

         }//for

    if (j > 0 && m < 7)  { Soil_Temperature = tj/j; }
    else {Soil_Temperature = -21.0;}

    if (Soil_Temperature > 0.0) {

    Serial.println("Heating␣...");
    digitalWrite(pin_Heater, HIGH);
    delay(Heating_Time);                          // Time of heating, ms
    digitalWrite(pin_Heater, LOW);

    Serial.println("Heat␣dissipation␣...");
    dtime = millis(); // start time of Heat dissipation

    t_current = (Soil_Temperature + 5.0);
    DS18B20_measure();
    if (m_err == 0) {t_current = celsius;}

    m_cycle = 0;

    while (t_current > (Soil_Temperature + 1.0) && m_cycle < 250)
        {
         DS18B20_measure();
         if (m_err == 0) {t_current = celsius;}
         Serial.println(t_current );
         m_cycle++;
        }//while

    Time_Heat_Dissipation = (millis() - dtime)/1000.0;
    Serial.print("Time␣of␣Heat␣Dissipation␣=␣");
    Serial.print(Time_Heat_Dissipation);
    Serial.println(",␣seconds");

   }//Soil_Temperature > 0

}// Measure_SMT
```

3.3.2 Test rain sensor[20]

Listing 2: Test rain sensor

```
   /*

All the resources for this project:
https://randomnerdtutorials.com/

*/

int rainPin = A0;
int greenLED = 6;
int redLED = 7;
// you can adjust the threshold value
int thresholdValue = 500;

void setup(){
  pinMode(rainPin, INPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  digitalWrite(greenLED, LOW);
  digitalWrite(redLED, LOW);
  Serial.begin(9600);
}

void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(rainPin);
  Serial.print(sensorValue);
  if(sensorValue < thresholdValue){
    Serial.println("␣-␣It's␣wet");
    digitalWrite(greenLED, LOW);
    digitalWrite(redLED, HIGH);
  }
  else {
    Serial.println("␣-␣It's␣dry");
    digitalWrite(greenLED, HIGH);
    digitalWrite(redLED, LOW);
  }
  delay(500);
}
```

3.3.3 Test motion control sensor[9]

Listing 3: Test motion control sensor

```
/*
  Ultrasonic Sensor HC-SR04 and Arduino Tutorial

  by Dejan Nedelkovski,
  www.HowToMechatronics.com

*/
// defines pins numbers
const int trigPin = 9;
const int echoPin = 10;
// defines variables
long duration;
int distance;
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an
     Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an
     Input
  Serial.begin(9600); // Starts the serial communication
}
void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel
     time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2;
  // Prints the distance on the Serial Monitor
  Serial.print("Distance:␣");
  Serial.println(distance);
}
```

3.3.4 Full design code

The Dramco-Uno library and the rest of the code can be found on the following GitHub-page: https://github.com/JGheysens/Design-monitoring-system.
For more information about the used LoRaWan-connectivity and how you need to connect to the IoT (Internet of Things), check the following page:
https://dramco.be/projects/dramco-uno/

Listing 4: Full design code

```
#include <Arduino.h>
/*
Design of an IoT system for monitoring chemical
and physical properties of soil (moisture,
temperature, pH), precipitation, pest intrusion
for pineapple and tomato crops

Developed by Jonathan Gheysens & Fidele Houeto

Example for DRAMCO-UNO

Connection wires of SMT-01 cable to DRAMCO UNO:
Yellow DS  - to Pin 5 (R2 4.7k - 2.0k to +5V, depending
    on length of the cable)
Green 2N   - to Pin 4 (via R1 (10k - 2.0k, depending on
    length of the cable)
Red        - to +5V
Black      - to GND

Connection wires of FC-37 rain sensor:
Blue  - to Pin A0
Red   - to +5V
Black - to GND

Connection wires of HC-SR04 sensor:
Pink - to pin 10 (trig)
Cyan - to pin 9 (echo)
Red  - to +5V
Black- to GND

Connection wires of MA40S4S ultrasonic transducer:
Purple  - to pin 8
Black- to GND
*/

//definitions and initialisations for DRAMCO-UNO board
```

```
#include <Dramco-UNO.h>

LoraParam DevEUI = "70B3D57ED005DD24";
LoraParam AppKey = "24444A42B9CCA5F41CAC4D9C3C0048F6";


//definitions and initialisations for SMT-01 sensor
#include <OneWire.h>

#define DARK  LOW
#define LIGHT HIGH
#define ON    HIGH
#define OFF   LOW

OneWire  ds(5);  // 1-Wire to Pin 5: sensor DS18B20
    connected

byte i;
byte present = 0;
byte type_s;
byte data[12];
byte addr[8];
float celsius;
int m_err;

int pin_Led = 13; //BUILTIN-LED
int pin_Heater = 4; //pin heating component: bipolar
    transistor 2N2222A

int DS_found = 0;
float Time_Heat_Dissipation, Soil_Moisture,
    Soil_Temperature;
unsigned long Heating_Time = 30000; //ms
int j, k;

void DS18B20_init(void);
void DS18B20_measure(void);
void Measure_SMT (void);

unsigned long mtime;
unsigned long set_mtime;


//definitions and initialisations for FC-37 sensor
int rainPin = A0;
int thresholdValue = 500; //adjust this value for your own desing

//definitions and initialisations for HC-SR04 sensor
int trigPin = 10;
int echoPin = 9;
long duration;
float distance;

//definitions and initialisations for MA40S4S ultrasonic transducer
#define MAX_DISTANCE 200 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.
int ultrasonicPin = 8;

//setup
void setup()
{
   pinMode(pin_Led,OUTPUT);
   pinMode(pin_Heater, OUTPUT);

   pinMode(rainPin, INPUT);

   pinMode(trigPin, OUTPUT);
   pinMode(echoPin, INPUT);

   digitalWrite(pin_Led,DARK);
   digitalWrite(pin_Heater, LOW);

//LORAWAN CONNECTIVITY
   DramcoUno.begin(DevEUI, AppKey);

// UART communication setup: easy to read out with serial monitor
   Serial.begin(9600);
   delay(10);
   Serial.println("");

   Serial.println("Initialization␣of␣DS18B20␣...␣");

   DS_found = 0;
   DS18B20_init();
   if (DS_found == 1){
       digitalWrite(pin_Led, LIGHT);
       Serial.println("Initialization␣is␣Ok");
```

```
      delay(1000);
      digitalWrite(pin_Led,DARK);
     }

  set_mtime = 1000;
  mtime = millis();

}//setup


void loop()
{ //rain sensor
  int rainValue = analogRead(rainPin);
  Serial.print(rainValue);
  if(rainValue < thresholdValue){
    Serial.println("␣-␣It’s␣raining");
  }
  else {
    Serial.println("␣-␣It’s␣not␣raining");
  }
  DramcoUno.sendRain(rainValue);

  // HC-SR04
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2;
  // Prints the distance on the Serial Monitor
  Serial.print("Distance:␣");
  Serial.println(distance);
  DramcoUno.sendUltrasonic(distance);

  //ultasonic wave
  if (distance<MAX_DISTANCE){//pest intrusion detected:
     reflection of ultrasonic wave by pest
      tone(ultrasonicPin, 65000, 1000);//ultrasonic wave
         at 65kHz for 1 second
  }

 if (millis() - mtime > set_mtime) {

  digitalWrite(pin_Led, LIGHT);
```

```

// Measurement

  if(DS_found == 1){

  Serial.println("Start␣measurement");
  Measure_SMT();

// Converting the Time of Heat Dissipation to Soil Moisture, %

// as example
  float Sensor_Dry = 250.0; //Time of Heat Dissipation for Dry Sensor
  float Sensor_Wet = 35.0;  //Time of Heat Dissipation for Wet Sensor

  Soil_Moisture = map(Time_Heat_Dissipation, Sensor_Dry, Sensor_Wet, 0.0, 100.0);
  if (Soil_Moisture < 0.0) Soil_Moisture = 0.0;
  if (Soil_Moisture > 100.0) Soil_Moisture = 100.0;

  Serial.print("Soil␣Moisture␣=␣");
  Serial.print(Soil_Moisture);
  Serial.println(",␣%");
  DramcoUno.sendSoilMoisture(Soil_Moisture);
  Serial.print("Temperature␣of␣Soil␣=␣");
  Serial.print(Soil_Temperature);
  Serial.println(",␣oC");
  DramcoUno.sendSoilTemperature(Soil_Temperature);
  }
  else{
   Serial.println("Next␣try␣to␣Initialization␣of␣DS18B20␣...␣");
   DS_found = 0;
   DS18B20_init();
   if (DS_found == 1){
       digitalWrite(pin_Led, LIGHT);
       Serial.println("Initialization␣is␣Ok");
       delay(1000);
       digitalWrite(pin_Led,DARK);
      }

    }

  set_mtime = 420000; // recommended pause between measurements, ms (7 minutes)
  mtime = millis();

  Serial.println("Waiting␣for␣the␣next␣measurement␣...");
  digitalWrite(pin_Led,DARK);

  }// if mtime

  delay(10);

}// loop


void DS18B20_init(void){ // --------------------------------------------------------------

if ( !ds.search(addr)) {
    DS_found = 0;
    Serial.println("Sensor␣not␣found.");
    Serial.println();
    ds.reset_search();
    delay(250);
    return;
  }//if

Serial.print("ROM␣=");
  for( i = 0; i < 8; i++) {
    Serial.write('␣');
    Serial.print(addr[i], HEX);
  }

  if (OneWire::crc8(addr, 7) != addr[7]) {
      Serial.println("CRC␣is␣not␣valid!");
      DS_found = 0;
      return;
  }
  Serial.println();

// the first ROM byte indicates which chip
  switch (addr[0]) {
    case 0x10:
      Serial.println("␣␣Chip␣=␣DS18S20");  // or old DS1820
      type_s = 1;
      break;
    case 0x28:
      Serial.println("␣␣Chip␣=␣DS18B20");
      type_s = 0;
      DS_found = 1;
      break;
    default:
      Serial.println("Device␣is␣not␣a␣DS18x20␣family␣device.");
      return;
  }//switch

}//DS18B20_init

void DS18B20_measure(void) {//------------------------------------------------------

  m_err = 0;

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1);    // start conversion, with parasite power on at the end

  delay(1000);          // time need for conversion

  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);       // Read Scratchpad

  for ( i = 0; i < 12; i++) {           // we need 12 bytes resolution
    data[i] = ds.read();
  }

 if (OneWire::crc8(data, 8) != data[8]) {
      Serial.println("CRC␣is␣not␣valid!");
      m_err = 1;
  }


  // Convert the data to actual temperature
  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3; // 9 bit resolution default
    if (data[7] == 0x10) {
      // "count remain" gives full 12 bit resolution
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    // at lower res, the low bits are undefined, so let's
        zero them
    if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution
        , 93.75 ms
    else if (cfg == 0x20) raw = raw & ~3; // 10 bit res,
        187.5 ms
    else if (cfg == 0x40) raw = raw & ~1; // 11 bit res,
        375 ms
    //// default is 12 bit resolution, 750 ms conversion
        time
  }

  celsius = (float)raw / 16.0;

 }//DS18B20_measure

void Measure_SMT ()
   {
    float t_current, tj;
    int m_cycle, j, m;
    unsigned long dtime;

    Time_Heat_Dissipation = 0.0;
    j = 0;
    m = 0;
    tj = 0.0;

    Serial.println("Temperature of Soil measurement ...")
        ;
    for (m_cycle = 0; m_cycle<10; m_cycle++)
        {
          DS18B20_measure();
          if (m_err == 0)
              {
               tj = tj + celsius;
               j++;
               Serial.println(celsius);
              }//if
          else {m++;}

         }//for

    if (j > 0 && m < 7)  { Soil_Temperature = tj/j; }
    else {Soil_Temperature = -21.0;}

    if (Soil_Temperature > 0.0) {

    Serial.println("Heating␣...");
    digitalWrite(pin_Heater, HIGH);
    delay(Heating_Time);                          // Time of
        heating, ms
    digitalWrite(pin_Heater, LOW);

    Serial.println("Heat␣dissipation␣...");
    dtime = millis(); // start time of Heat dissipation

    t_current = (Soil_Temperature + 5.0);
    DS18B20_measure();
    if (m_err == 0) {t_current = celsius;}

    m_cycle = 0;

    while (t_current > (Soil_Temperature + 1.0) &&
        m_cycle < 250)
         {
          DS18B20_measure();
          if (m_err == 0) {t_current = celsius;}
          Serial.println(t_current );
          m_cycle++;
         }//while

    Time_Heat_Dissipation = (millis() - dtime)/1000.0;
    Serial.print("Time␣of␣Heat␣Dissipation␣=␣");
    Serial.print(Time_Heat_Dissipation);
    Serial.println(",␣seconds");

   }//Soil_Temperature > 0

}// Measure_SMT
```

3.4 Analysis

Table 6: Cost analysis: full design

Description	**Price ($)**
SMT-01 (soil moisture & temperature sensor)	5
FC-37 (resistive rain sensor)	3
HC-SR04 (ultrasonic sensor: pest detection)	5
MA40S4S (ultrasonic transducer: pest repeller)	9
Dramco-Uno (microcontroller board)	10
Manufacturing & placement	Hard to estimate
Total	**32**

This design is just a prototype and needs to be tested before deploying this on a big scale. We can still make some improvements in the code by executing the different measurements on different cores of the micro-controller. Like this, we can read out the different sensors at the same time. We can also include a sleep-function, where we set a delay so we don't take measurements the whole day. This will also reduce the power consumption of our system.

4 Conclusion

In this report we discussed all the types of sensors we need to design an IoT-sytem to monitor the chemical and physical properties of soil (moisture, temperature and pH), precipitation and pest intrusion. Additionally, a prototype design has been developed, accompanied by the corresponding code implementation.

References

[1] Guide for soil moisture sensor yl-69 or hl-69 with arduino, 2023. https://randomnerdtutorials.com/guide-for-soil-moisture-sensor-yl-69-or-hl-69-with-the-arduino/.

[2] Sku:sen0193, 2023. https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193.

[3] Soil moisture temperature sensor, 2023. https://wiki.seeedstudio.com/Sensor/SenseCAP/SenseCAP$_Probe/Soil - Moisture - Temperature - Sensor/$.

[4] Soil ph sensor, 2023. https://www.renkeer.com/product/soil-ph-sensor/.

[5] Soil temperature sensor probes, 2023. https://www.vegetronix.com/Products/THERM200/.

[6] Vh400 low-cost soil moisture sensors, 2023. https://vegetronix.com/soil-moisture-sensor.

[7] Arduino. tone(), 2023. https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/.

[8] Hydreon Corporation. Rain gauge model rg-11, 2023. https://rainsensors.com/products/rg-11/.

[9] Dejan. Ultrasonic sensor hc-sr04 and arduino – complete guide, 2015. https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/.

[10] DFROBOT. Pir motion sensor v1.0 sku:sen0171, 2017. https://wiki.dfrobot.com/PIR$_Motion_Sensor_V1.0_SKU_SEN0171$.

[11] Dragino. Lsph01 – lorawan soil ph sensor, 2022. https://www.dragino.com/products/agriculture-weather-station/item/184-lsph01.html.

[12] Conrad Electronic. Iduino for maker's life pir motion sensor (se062), 2017. https://asset.conrad.com/media10/add/160267/c1/-/en/001485335DS01/fiche-technique-1485335-capteur-de-mouvements-pir-iduino-se062-1485335-1-pcs.pdf.

[13] Greensensors. Smt soil moisture sensor for arduino, 2022. https://github.com/greensensors/SMT-Soil-Moisture-Sensor-for-Arduino.

[14] Jianjing Huang. Grove - moisture sensor, 2023. https://wiki.seeedstudio.com/Grove-Moisture$_Sensor/$.

[15] Hydreon. Hydreon solid-state rain sensors, 2023. https://rainsensors.com/.

[16] Ian L. Pepper Janick F. Artiola, Mark L. Brusseau. Environmental monitoring and characterization, 2004. https://www.sciencedirect.com/book/9780120644773/environmental-monitoring-and-characterization.

[17] Toni Klopfenstein. Soil moisture sensor, 2015. https://github.com/sparkfun/Soil$_{M}oisture_{S}ensor.$

[18] Murata. Ma40s4s, 2023. https://www.murata.com/en-us/products/productdetail?partno=MA40S4S.

[19] BC Robotics. Tipping rain gauge, 2023. https://bc-robotics.com/shop/tipping-rain-gauge/.

[20] Rui. Guide for rain sensor fc-37 or yl-83 with arduino, 2018. https://randomnerdtutorials.com/guide-for-rain-sensor-fc-37-or-yl-83-with-arduino/.

[21] Shawn. Soil moisture sensor – getting started with arduino, 2020. https://www.seeedstudio.com/blog/2020/01/10/what-is-soil-moisture-sensor-and-simple-arduino-tutorial-to-get-started/.

[22] Sinotronics. How to use a soil ph meter (and why it matters), 2020. https://www.instrumentchoice.com.au/news/how-to-use-a-soil-ph-meter-and-why-it-matters.

[23] Sinotronics. Rain gauges – what types are the main types, and what are the benefits?, 2020. https://www.instrumentchoice.com.au/news/rain-gauges-what-types-are-the-main-types-and-what-are-the-benefits.