```
#include <SPI.h>
#include <MFRC522.h>
#include <WiFi.h>
#include <PubSubClient.h>

#define SS_PIN 4
#define RST_PIN 25
MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance.

#include "DHT.h"
#define DHTPIN 33
#define DHTTYPE DHT11  // DHT 11
DHT dht(DHTPIN, DHTTYPE);

#include <OneWire.h>
#include <DallasTemperature.h>
const int oneWireBus = 32;
OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);

#include <LiquidCrystal_I2C.h>
int lcdColumns = 16;
int lcdRows = 2;
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);
```

We importeren de nodige bibliotheken voor het communiceren met de sensoren, het WiFi-netwerk en het MQTT-protocol, we definiëren ook de pinnen waarop de sensoren zijn aangesloten. We stellen een LCD scherm in. Voor adres en de grootte van het scherm.

```
const char* ssid = "********";
const char* password = "*********";

// MQTT instellingen
const char* mqtt_server = "192.168.*.***";
const int mqtt_port = 1883;
const char* mqttUser = "******";
const char* mqttPassword = "******";
```

We configureren de instellingen om verbinding te maken met een Wifi-netwerk en een MQTT-broker. We definieren de naam en wachtwoord van het Wifi-netwerk en ook het IP-adres, poort, en inloggegevens van de MQTT-broker.

```
WiFiClient espClient;
PubSubClient client(espClient);
```

We maken een client aan om verbinding te maken met het wifi-netwerk en de MQTT-broker.

```
const int pomp = 16;  //pomp
const int vent = 13;  //vent
const int trigPin = 27;
const int echoPin = 26;
```

```
long duration, cm;

float LT;  //lucht temp
float LV;  //lucht Hum
float BT;  //bodem Temp
float BV;  //bodem Hum

float WP;  //water pomp status
float V;   //ventilator status

float ds18b20 = sensors.getTempCByIndex(0);
float BVsensor = map(analogRead(35), 0, 4095, 100, 0);
```

We definieren variabelen voor het aansturen van een waterpomp, een ventilator, het meten van de afstand met een ultrasone sensor en het verzamelen van temperatuur voor zowel lucht als bodem.

```
void setup() {
  Serial.begin(115200);  // Initiate a serial communication

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

  lcd.init();
  lcd.backlight();
  SPI.begin();         // Initiate  SPI bus
  mfrc522.PCD_Init();  // Initiate MFRC522
  dht.begin();
  sensors.begin();

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(5000);
    Serial.println("Verbinding maken met WiFi...");
  }

  Serial.println("Verbonden met WiFi");
  Serial.print("IP-adres: ");
  Serial.println(WiFi.localIP());

  client.setServer(mqtt_server, mqtt_port);
}
```

In de void setup() starten we de seriële communicatie met de seriële monitor. We stellen onze pinmode's in zodat we informatie kunnen lezen en ook apparaten aansturen. We starten externe apparaten zoals onze lcd, rfid, … vervolgens maken we verbinding met het Wifi-netwerk eens dat gebeurd is gaan we verbinding maken met de MQTT-broker.

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();
```

We kijken na of de MQTT-client nog altijd verbonden is met de broker.

```
  sensors.requestTemperatures();
  float ds18b20 = sensors.getTempCByIndex(0);
  float BVsensor = map(analogRead(35), 0, 4095, 100, 0);

  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  cm = (duration / 2) / 29.1;

  float afstand = map(cm, 8, 28, 100, 0);
```

We voeren verschillende metingen uit waaronder die van de ds18b20. Voor onze bodemvochtigheids sensor gebruiken we een map functie zodat de waarde tussen 0 en 100 zit. Vervolgens stellen we een ultrasone sensor in om de afstand te geven in centimeters. Vervolgens maken we een extra variabele voor te weten hoeveel procent er in onze water tank dus 8cm komt overeen met 100% en 28cm met 0%.

```
if (cm < 22) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(dht.readHumidity());
    lcd.print(" %");

    lcd.setCursor(9, 0);
    lcd.print(dht.readTemperature());
    lcd.print(" C");

    lcd.setCursor(0, 1);
    lcd.print(BVsensor);
    lcd.print(" %");

    lcd.setCursor(9, 1);
    lcd.print(ds18b20);
    lcd.print(" C");
```

```
  } else if (cm > 22) {
    lcd.clear();
    lcd.setCursor(1, 0);
    lcd.print("Refill Water!");
    delay(2000);
  }
```

We geven hier aan dat wanneer de ultrasone sensor een afstand meet kleiner dan 22cm. Dan zullen onze waardes op het lcd scherm geprint worden. Als de afstand groter is dan 22cm dan zal er “Refill water” geprint worden op het lcd scherm.

```
  Serial.println(dht.readHumidity());
  Serial.println(dht.readTemperature());
  Serial.println(BVsensor);
  Serial.println(ds18b20);
  Serial.print(afstand);
  Serial.print(" %  ");
  Serial.print(cm);
  Serial.println(" cm");
```

We zorgen ervoor dat de gemeten waarden van de sensoren geprint worden in de seriele monitor.

```
String payload1 = String(dht.readTemperature());
  String payload2 = String(dht.readHumidity());
  String payload3 = String(ds18b20);
  String payload4 = String(BVsensor);
  String payload5 = String(LT);
  String payload6 = String(LV);
  String payload7 = String(BT);
  String payload8 = String(BV);
  String payload9 = String(V);
  String payload10 = String(WP);
  String payload11 = String(afstand);
```

We zetten de gemeten sensorwaarden en statussen om naar strings zodat we deze via MQTT kunnen doorsturen.

```
client.publish("serre/dht11/Temperature", payload1.c_str());
  client.publish("serre/dht11/Humidity", payload2.c_str());
  client.publish("serre/ds18b20/bodemtemp", payload3.c_str());
  client.publish("serre/BVsensor/bodemvocht", payload4.c_str());
  client.publish("serre/LT/Gevraagde lucht temp", payload5.c_str());
  client.publish("serre/LV/Gevnraagde lucht hum", payload6.c_str());
  client.publish("serre/BT/gevraagde bodem temp", payload7.c_str());
  client.publish("serre/BV/gevraagde bodem hum", payload8.c_str());
  client.publish("serre/Ventilator/Status vent", payload9.c_str());
  client.publish("serre/Water pomp/Status pomp", payload10.c_str());
  client.publish("serre/Water tank/Water niveau", payload11.c_str());
```

We publiceren de sensorwaarden en statussen naar verschillende MQTT-topics. Elke topic staat voor een specifieke sensorwaarde.

```
  if (BVsensor < BV - 3) {  //water pomp
    WP = 1;
    delay(2000);
    digitalWrite(pomp, HIGH);
    delay(5000);
  } else if (BVsensor > BV + 3) {
    WP = 0;
    digitalWrite(pomp, LOW);
  }
```

We regelen hier de waterpomp op basis van de gemeten waarde van de bodemvochtigheids sensor (BVsensor) en de ingestelde waarde BV met een speling van 3%. Dus wanneer de sensorwaarde onder de gevraagde waarde BV zit met een 3% speling dan gaat de pomp aan. Wanneer deze boven de gevraagde waarde zit gaat de pomp uit. We geven hier ook mee wat de status is van de pomp dus 1 is aan 0 is uit.

```
if (dht.readTemperature() > LT + 3) {  //Ventilator
    V = 1;
    digitalWrite(vent, HIGH);
  } else if (dht.readTemperature() < LT - 3) {
    V = 0;
    digitalWrite(vent, LOW);
  }
```

Dit is exact hetzelfde als bij de water pomp we regelen hier op basis van de lucht temperatuur en geven ook de status door.

```
  // Select one of the cards
  if (!mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  //Show UID on serial monitor
  Serial.print("UID tag :");
  String content = "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }

  Serial.println();
  Serial.println("Message : ");
  content.toUpperCase();
```

Hierdoor lezen we de RFID-kaart en tonen we de ID van de kaart.

```
  if (content.substring(1) == "63 3F 55 14") {  //Tijm
    LT = 24;
    LV = 65;
    BT = 20;
    BV = 15;
  }
  if (content.substring(1) == "A3 37 CE 26") {  //Paprika
    LT = 21;
    LV = 65;
    BT = 20;
    BV = 30;
  }
```

Er zijn specifieke badges gelinkt aan bepaalde waardes dus bv de eerste is voor tijm. We stellen hierdoor onze gevraagde lucht temperatuur/ vochtigheid en bodem temperatuur/vochtigheid.

```
void reconnect() {
  while (!client.connected()) {
    Serial.println("Verbinding maken met MQTT Broker...");

    if (client.connect("DHT11Client", mqttUser, mqttPassword)) {
      Serial.println("Verbonden met MQTT Broker");
    } else {
      Serial.print("Fout bij het verbinden met MQTT Broker. Foutcode: ");
      Serial.print(client.state());
      Serial.println(" Opnieuw proberen");
    }
  }
}
```

De reconnect functie maakt continu verbinding met de broker tot dat het lukt en drukt verbindingsstatusberichten af voor de seriële monitor.