Politechnika Wrocławska

Zastosowania informatyki w Gospodarce

Stacja automatycznego podlewania upraw

Autorzy:
Tomasz Dylak
Damian Strycharczuk
Damian Cywiński
Marcin Cieślak

Prowadzący Zajęcia:
Dr Inż. Marek Woda

HR EXCELLENCE IN RESEARCH

Politechnika Wrocławska

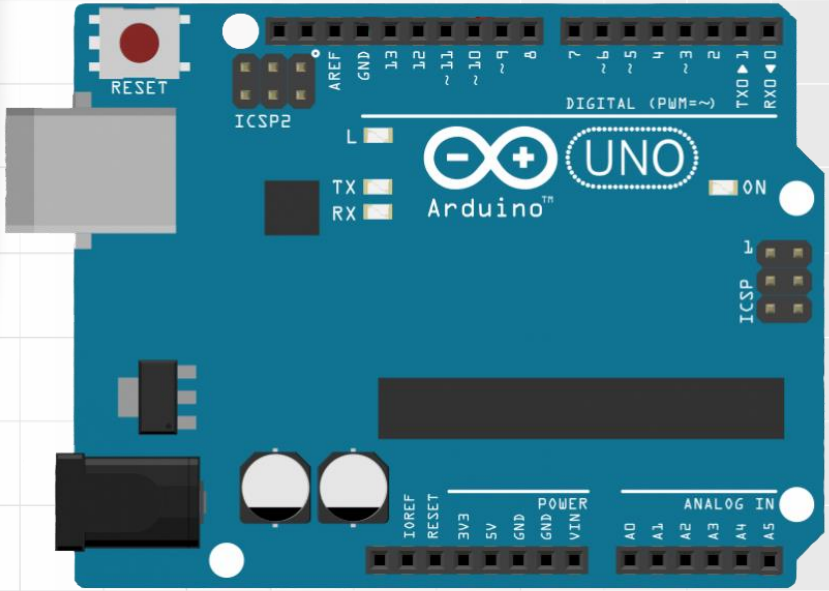
MODUŁ ARDUINO

Dokumnetacja

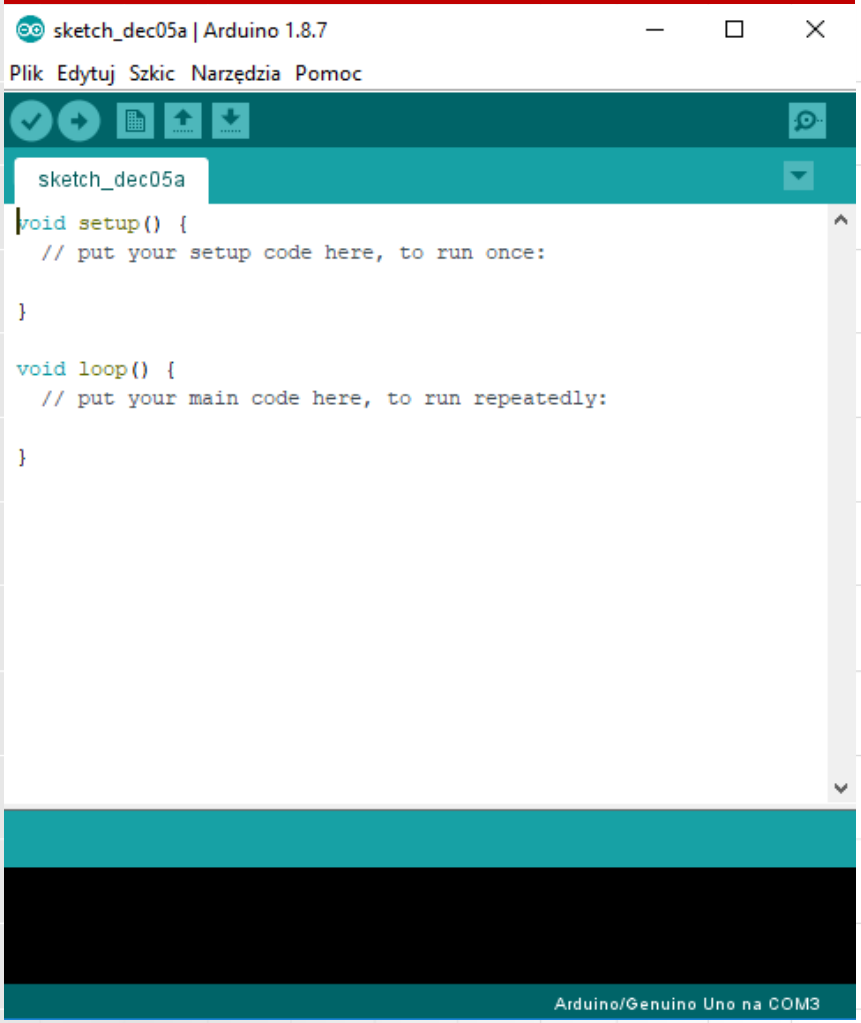
Arduino

Arduino UNO

- Procesor – ATmega328,
- 32 kB pamięci Flash,
- 2 kB pamięci RAM,
- 14 cyfrowych wejść/wyjść
- 6 wejść analogowych



Programowanie Arduino



Zastosowane czujniki

DHT22

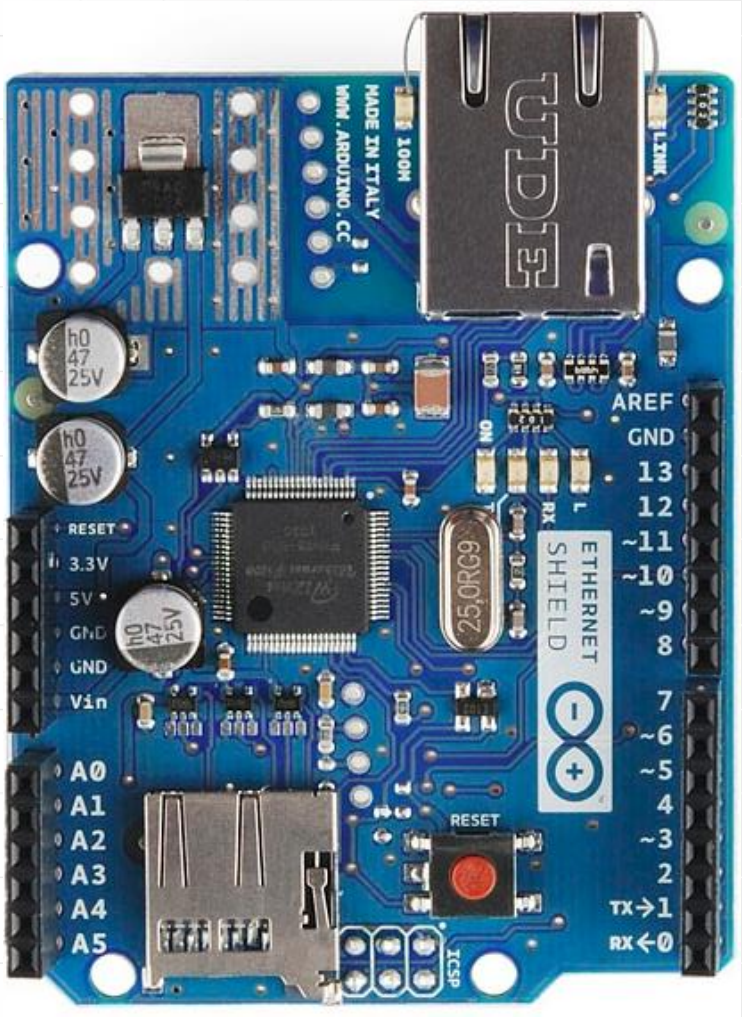
- Pomiar temperatury i wilgotności,
- Zakres temperatury od -40 do 80[℃],
- Dokładność pomiaru temperatury 0.5[℃],
- Zakres wilgotności od 0 do 100[%],
- Dokładność pomiaru wilgotności 2[%].

Czujnik wilgotności gleby

- Pomiar wilgotności gleby
- Pomiar wilgotności – wartość analogowa oparta na rezystancji
- Wyjście cyfrowe z regulowanym poziomem wyzwalania
- Używany zakres – konwersja na 0 do 100 [%]

Komunikacja stacji z komputerem

Powłoka Ethernet



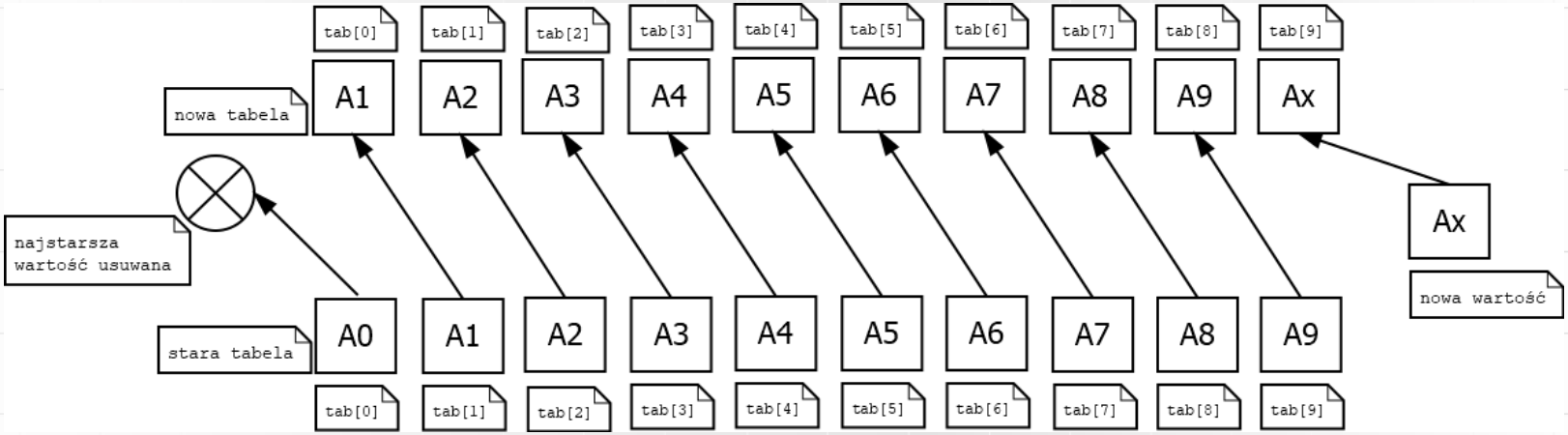
- Powszechne zastosowanie wtyczki Rj-45,
- Większe odległości pomiędzy stacją a serwerem,
- Trwalsza konstrukcja,
- Uniezależnienie stacji od zasilania z komputera.

Załączanie pompy

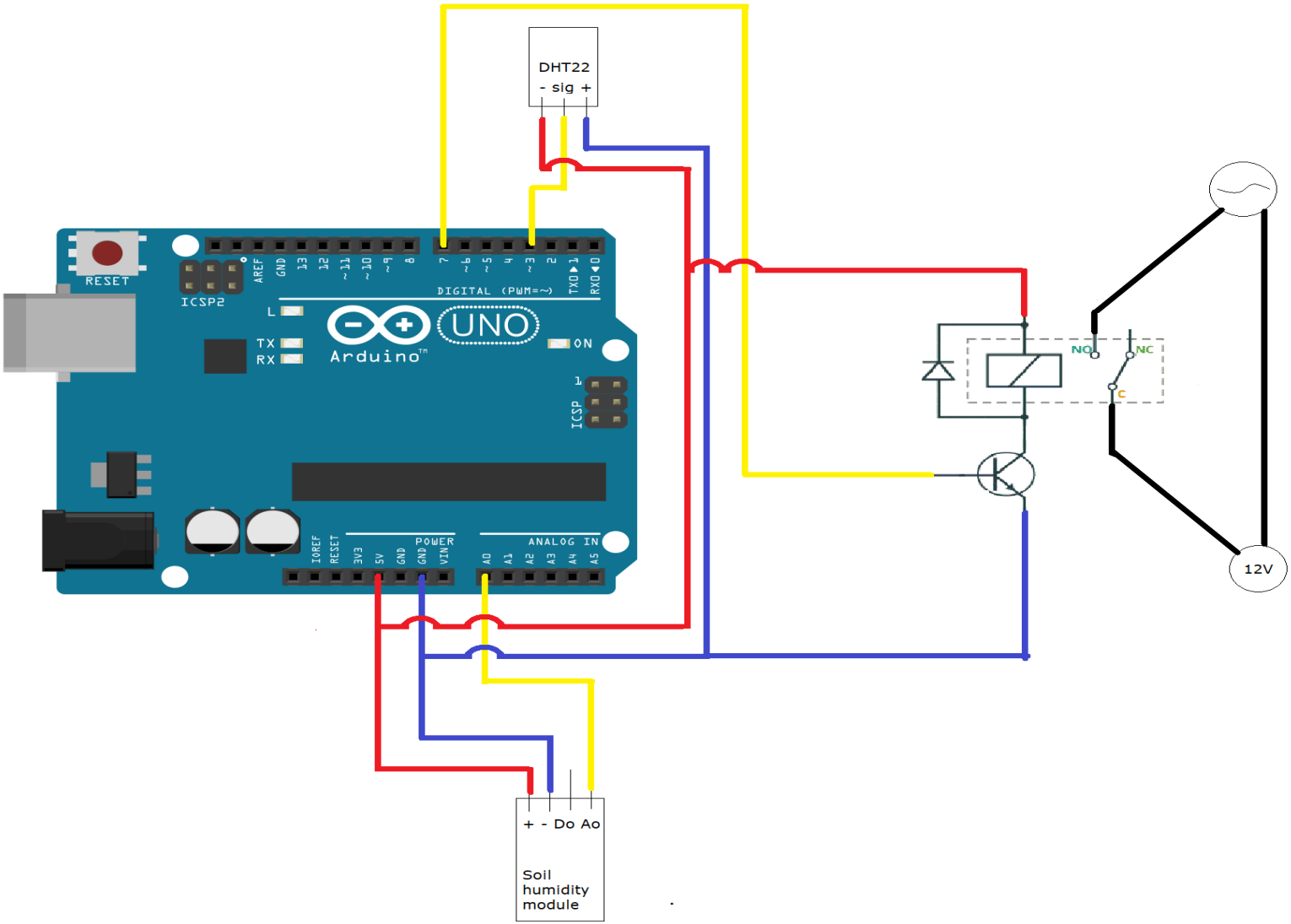
Osprzęt do podlewania

- Moduł przekaźnika 1 kanał, 7A/240VAC, cewka 5V
- Zasilacz sieciowy, wejście AC 100-240V, 24W, 50/60 Hz, wyjście DC 12V 2A
- Pompa do cieczy 280l/h, 12V DC, pobór prądu 4,5W, wysokość pompowania 300cm, żywotność 30000h

Rejestr przesuwny



Schemat systemu



Implementacja kodu

- Konfiguracja Ehternet

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,137,200); //IP address for your arduino.
char server[] = "192.168.137.1"; //IP address of your server.
EthernetClient client;
```

Implementacja kodu

- Konfiguracja czujników

```
#define DHTPIN 3
#define AO A0

// DHT22 configuration
#define DHTTYPE    DHT22     // DHT 22 (AM2302)
DHT_Unified dht(DHTPIN, DHTTYPE);
```

Implementacja kodu

- Zmienne globalne

```
//global variables
int id_ard = 1;              // id arduino
double temp;                 // temperatura odczytywana z DHT22
double soil;                 // wilgotność gleby
double hum;                  // wilgotność powietrza odczytywana z DHT22
double tab_temp[10];         // tablica z 10 ostatnimi pomiarami dla temperatury
double tab_soil[10];         // tablica z 10 ostatnimi pomiarami dla wilgotności gleby
double tab_hum[10];          // tablica z 10 ostatnimi pomiarami dla wilgotności powietrza
double temp_sr;              // temperatura uśredniona
double soil_sr;              // wilgotność gleby uśredniona
double hum_sr;               // wilgotność powietrza uśredniona
double pumplong;             // jak długo pompa będzie podlewać
double pumptimes;            // jak często kwiatki będą podlewane
long double now;             // jaki jest obecnie czas
String data;                 // zmienna przechowująca dane do wysyłania POST'em na serwer
String rcv="";               // zmienna pomocnicza do odczytywania danych z serwera
double min_temp;             // minimalna temperatura
double min_hum;              // minimalna wilgotność powietrza
double min_soil;             // minimalna wilgotność gleby
int alert;                   // zmienna informująca czy było jakieś zawiadomienia
```

Implementacja kodu

- Funkcja *void setup()*

```
Serial.begin(9600);                 // wystartowanie monitora portu szeregowego
Serial.println("Inicjalizacja!");
Ethernet.begin(mac, ip);            // start komunikacji Ethernet
pinMode(A0, INPUT);                 // ustawienie pinu A0 jako wejście
pinMode(7, OUTPUT);                 // ustawienie pinu 7 jako wyjście
dht.begin();                        // start czujnika DHT22
 delay(15000);                      // opóźnienie by Ethernet i czujniki zdążyły się włączyć
 // sensor_t sensor;

data = "";
min_soil=1;                        // domyślna wartość minimalna dla wilgotności gleby
min_temp=0;                        // domyślna wartość minimalna dla temperatury
min_hum=0;                         // domyślna wartość minimalna dla wilgotności powietrza
pumptimes=259200;                  // domyślna wartość dla czasu następnego podlewania
pumplong=10;                       // domyślna wartość dla długości podlewania
alert=0;
```

Implementacja kodu

- Zastosowane biblioteki

#include <Adafruit_Sensor.h> // biblioteka obsługująca czujniki,

#include <DHT.h> // biblioteka ogługująca moduł czujnika DHT22,

#include <DHT_U.h> // biblioteka ogługująca moduł czujnika DHT22,

#include <SPI.h> // synchroniczny protokół danych szeregowych,

#include <Ethernet.h> // umożliwia podłączenie karty Arduino do internetu, może służyć jako serwer lub jako klient

#include <Wire.h> // umożliwia komunikację z urządzeniami I2C/TWi (w obecnej implementacji nie jest ona używana).

Implementacja kodu

- Funkcja *void setup()*

```
// tworzenie pierwszej tablicy do rejestru przesuwnego
sensors_event_t event;
 do{
 //sensors_event_t event;
 dht.temperature().getEvent(&event);        //pobieranie temperatury
 temp=event.temperature;
 dht.humidity().getEvent(&event);           //pobieranie wilgotności powietrza
 hum=event.relative_humidity;
 }while ((isnan(event.temperature))&&(isnan(event.relative_humidity)));
 soil=(1023 - analogRead(AO))/7;            //pobieranie wilgotności gleby

for (int i=0; i<=9; i++)
 {
    tab_temp[i]=temp;
    tab_soil[i]=soil;
    tab_hum[i]=hum;
  }
```

Implementacja kodu

- Funkcja *void watering()*

```
// jeżeli obecny czas jest większy lub równy cyklowi podlewania
if (now>=pumptimes)
  {
     digitalWrite(7, HIGH);          // włączenie pompy
     delay (pumplong*1000);          // czekanie
     digitalWrite(7, LOW);           // wyłączenie pompy
     now=0;                          // reset czasu
    }
```

Implementacja kodu

- Funkcja *void httpRequest()*

```
Serial.println("czytanie danych");
 Serial.println();
 if (client.connect(server, 80))
 {
  Serial.println("Connection established 1");
  client.print(String("GET ") + "/tryjson.php/?id_ard=" + id_ard + " HTTP/1.1\r\n" + "Host: " + server + "\r\n" + "Connection: close\r\n\r\n"); //GET request for server response.
  unsigned long timeout = millis();
  while (client.available() == 0)
  {
   if (millis() - timeout > 25000) //If nothing is available on server for 25 seconds, close the connection.
   {
    return;
    now = now + 25;
   }
  }
```

Implementacja kodu

- Funkcja *void httpRequest()*

```
String line = client.readStringUntil('\r'); //czytaj odpowiedź linia po linii
…
 line = client.readStringUntil('\r'); // czytaj linie do momentu uzyskania danych – 9ta linia

for (int i=3; i<line.length();i++)
 {
  if (line[i]=='a'){pumplong=rcv.toDouble(); rcv="";}                  // odczytaj długość podlewania
  else if (line[i]=='b'){pumptimes=rcv.toDouble()*3600; rcv="";}  // odczytaj cykl podlewania
  else if (line[i]=='c'){min_soil=rcv.toDouble(); rcv="";}              // odczytaj min wilgotność gleby
  else if (line[i]=='d'){min_temp=rcv.toDouble(); rcv="";}               // odczytaj min temperaturę
  else if (line[i]=='e'){min_hum=rcv.toDouble(); rcv="";}                // odczytaj min wilgotność pow.
  else rcv=rcv+line[i];
   }
  rcv="";
client.stop(); // Close the connection.
 }
 else { Serial.println("Connection failed 1"); }
```

Implementacja kodu

- Funkcja *void Loop()*

```
 httpRequest(); watering() ;                    // wywołaj funkcję do odczytania danych i podlewania
sensors_event_t event;                          // stwórz zmienną sensors_even_t
  dht.temperature().getEvent(&event);           // pobierz wartość temperatury z DHT22
     if (isnan(event.temperature)) {            // sprawdzanie czy temperatura się pobrała
     Serial.println(F("Error reading temperature!")); }
     else {
     temp=event.temperature;                    // przypisanie pomiaru do zmiennej
     Serial.print(F("Temperature: "));
     Serial.print(temp);
     Serial.println(F("°C")); }
    dht.humidity().getEvent(&event);            // pobierz wartość wilgotności z DHT22
    if (isnan(event.relative_humidity)) {       // sprawdzanie czy wilgotność się pobrała
    Serial.println(F("Error reading humidity!")); }
    else {
    hum=event.relative_humidity;                // przypisanie wilgotności do zmiennej
    Serial.print(F("Humidity: "));
    Serial.print(hum);
    Serial.println(F("%")); }
    Serial.print(F("Soil: "));
    soil=(1023 - analogRead(AO))/7;             // // pobierz wartość wilgotności gleby
    Serial.print(soil);
    Serial.println(F(" %"));
```

Implementacja kodu

- Funkcja *void Loop()*

```
double tmp1=0, tmp2=0, tmp3=0;
 for (int i=0; i<9; i++)    // usuń najstarszy pomiar z tabel
 { tmp1=tmp1+tab_temp[i+1];          //dodaj wszystkie wartości poza najstarszym
   tmp2=tmp2+tab_soil[i+1];
   tmp3=tmp3+tab_hum[i+1];
   tab_temp[i]=tab_temp[i+1];        //przesuń wszystkie pomiary „w dół” tablicy
   tab_soil[i]=tab_soil[i+1];
   tab_hum[i]=tab_hum[i+1]; }

tab_temp[9]=temp;   tab_soil[9]=soil; tab_hum[9]=hum;        // dodaj na koniec najnowszy pomiar
tmp1=tmp1+temp;   tmp2=tmp2+soil;   tmp3=tmp3+hum;        // dodaj ten pomiar do sumy
temp_sr=tmp1/10;  soil_sr=tmp2/10;  hum_sr=tmp3/10;        // wylicz średnią

 Serial.print(F("Temperature: "));
 Serial.print(temp_sr);
 Serial.println(F("°C"));
 Serial.print(F("Humidity: "));
 Serial.print(hum_sr);
 Serial.println(F("%"));
 Serial.print(F("Soil: "));
 Serial.print(soil_sr);
 Serial.println(F("%"));
```

Implementacja kodu

- Funkcja *void Loop()*

```
if (soil_sr<=min_soil)      // jeżeli średnia wilgotność gleby jest mniejsza od minimalnej
 { Serial.println();
   Serial.println("Uwaga! Niski poziom wilgotności gleby.");
   digitalWrite(7, HIGH);  // włącz pompę
   while (soil<=min_soil)
   { soil=(1023 - analogRead(AO))/7;
     delay(1000); }
   delay (pumplong*1000);  // poczekaj jeszcze chwilę
   digitalWrite(7, LOW);   // wyłącz pompę
   alert=1;
   now = 0; }            // zresetuj czas
// to samo dla minimalnej temperatury
  if (temp_sr<=min_temp)
 { Serial.println();
   Serial.println("Uwaga! Niska temperatura.");
   alert=1; }
   // I dla minimalnej wilgotności powietrza
  if (hum_sr<=min_hum)
 { Serial.println();
   Serial.println("Uwaga! Niska wilgotność powietrza.");
   alert=1; }
```

Implementacja kodu

- Funkcja *void Loop()*

```
// Tworzenie daty dla POST
 data = "temp=" + String(temp_sr) + "&soil=" + String(soil_sr) + "&humidity=" + String(hum_sr) + "&id_ard=" + String(id_ard) + "&alert=" + String(alert);
// Wysyłanie danych do bazy
 if (client.connect(server, 80)) //(server ip, port) – łączenie się z serwerem
  { Serial.println("Sending data to danedobazy2.php");
    client.print("POST /danedobazy2.php"); //GET request to write data to the database.
    client.println(" HTTP/1.1");
    client.println("Host: 192.168.137.1");
    client.println("Connection: close");
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.print("Content-Length: ");
    client.println(data.length());
    client.println();
    client.print(data);
    client.stop();
    Serial.println("Data sent");
    }
    else { Serial.println("Connection failed danedobazy.php");}
```

Implementacja kodu

- Funkcja *void Loop()*

```
delay(20000);        // odczekanie przed kolejnym pomiarem
now = now + 20;      // zwiększenie zmiennej czasu
alert=0;
```

Implementacja kodu

- *Danedobazy2.php*

```
<?php
        $id_arduino= $_POST["id_ard"];
        $temp= $_POST["temp"];
        $soil= $_POST["soil"];
        $humidity= $_POST["humidity"];
        $alert= $_POST["alert"];
        $API_URL = 'https://plants.ml/api/measurement/';
        $data = array(
        'id_arduino' => $id_arduino,
        'temp' => $temp,
        'soil' => $soil,
        'humidity' => $humidity
        );
 $string = http_build_query($data, '', '&');
$ch = curl_init($API_URL);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $string);
$result = curl_exec($ch);
curl_close($ch);
?>
```

Implementacja kodu

- *tryjson.php*

```
<?php
$API_URL = 'https://plants.ml/api/arduino/';
$id_ard = intval($_GET['id_ard']);
$curl = curl_init();
curl_setopt_array($curl, [
        CURLOPT_RETURNTRANSFER => 1,
        CURLOPT_URL => $API_URL.'/'.$id_ard.'/'
]);
$response = curl_exec($curl);
$response = json_decode($response, true);
curl_close($curl);

echo
json_encode("x".$response['how_long_to_water']."a".$response['how_often_to_water']."b".$response['min_temp']."c".$response['min_humidity']."d".$response['min_soil']."e");
?>
```