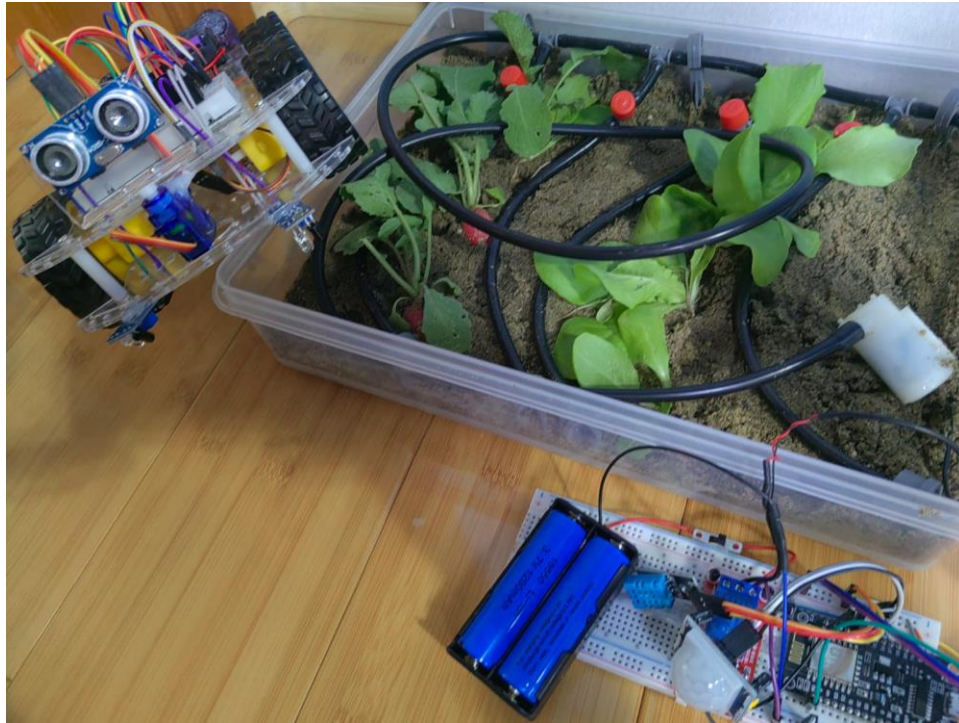


PREZENTAREA PROIECTULUI

„SMART PLANT”



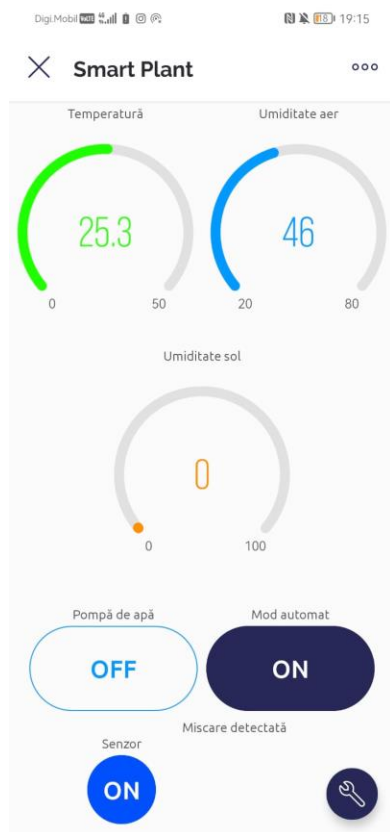
Smart Plant este un proiect care, dacă va fi adus la scară mare, va face ca domeniul agriculturii să fie în totalitate autonom.

- ***Capitolul I: Ce este Smart Plant (și care este utilitatea sa practică)?***

Smart Plant este un sistem automat de irigație, de monitorizare a plantelor și de afânare a solului.

Prin intermediul aplicației mobile, umiditatea și temperatura aerului, dar și umiditatea solului sunt monitorizate în timp real. De asemenea, irigația poate fi pornită și oprită manual, din butonul destinat în momentul în care modul automat este oprit. Aplicația va

trimite utilizatorului o notificare când simte că solul este uscat. Când modul automat este activ, irigația va fi pornită independent atunci când solul este uscat. Această obținere a fost introdusă în ideea în care se dorește ca udatul să fie făcut doar la anumite ore (dimineața sau seara, când apa este la temperatura potrivită). Este integrat și un senzor de mișcare care notifică utilizatorul atunci când un posibil animal se află prin apropiere, senzor care poate fi oprit în cazul în care.



interfața aplicației

Un buton fizic este integrat pentru pornirea și oprirea apei, în ideea în care accesul la internet ar putea fi oprit în anumite momente.

Tractorul inteligent urmează un traseu pe suprafața plantației, având rolul de a afâna solul. Acesta are capacitatea de a evita obstacolele care ar putea apărea în calea sa. Prototipul prezentat pornește și oprește sistemul de afânare în momentul în care observă un

obstacol, în ideea în care se dorește curățarea zonei din jurul unei plante. Acest aspect poate fi modificat prin cod în funcție de nevoile utilizatorului.

Întreg sistemul eficientizează domeniul agricol, putând fi controlat de oriunde din lume. El ușurează treaba fermierilor, reducând la zero munca fizică.

- **Capitolul II: Mecanică**

Sistemul funcționează pe baza a 7 motoare:

- 4 dintre ele sunt utilizate pentru deplazarea tractorului;
- unul rotește stânga-dreapta senzorul cu ultrasunete pentru a se asigura că niciun alt obstacol nu se afla în apropiere atunci când realizează manevra de depășire;
- unul este folosit pentru sistemul de afânare aflat în spatele tractorului;
- iar ultimul se află în interiorul pompei.

Proiectul este gândit astfel încât să consume cât mai puțină energie. Un exemplu pentru acest aspect este releul utilizat pentru pornirea și oprirea pompei de apă. La acesta este folosit port-ul NO, astfel încât să consume energie doar atunci când pompa funcționează (din motive evidente, pompa va sta mai mult timp închisă).

Pentru alimentare au fost folosiți acumulatori reutilizabili, care pot fi foarte ușor încărcăți de la surse sustenabile de energie (Ex: un încărcător solar).

- **Capitolul III: Electronică**

Tractorul are la bază o plăcuță de dezvoltare de tip Arduino UNO, perfectă pentru astfel de proiecte are numeroși pini atât pentru semnal de tip IN (senzori), cât și pentru semnal de tip OUT (motoare).

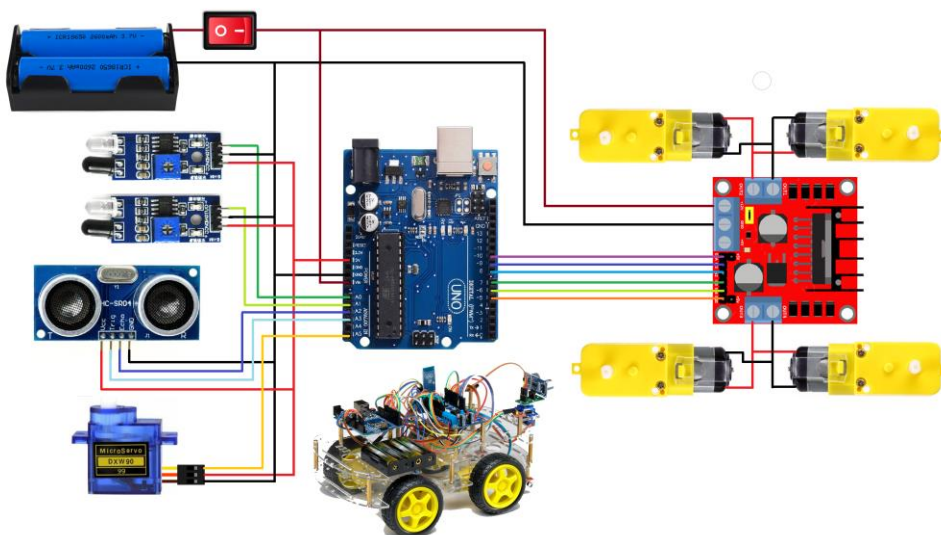
Motoarele sunt controlate de un driver de tip L298N, conectat la plăcuța Arduino, ele fiind eficient conectate între ele două câte două. Acest driver putea fi înlocuit cu un shield de motoare pentru Arduino, însă astfel s-ar fi limitat numărul de pini disponibili pentru restul componentelor.

Pentru detectarea obstacolelor este folosit un senzor cu ultrasunete de tip HC-SR04, el fiind rotit stânga-dreapta de un servo motor sg90, un motor cu posibilitate de control a sensului și timpului de rotire.

Marcajul de pe jos este urmărit cu ajutorul a doi senzori cu infraroșu. Linia neagră fiind în contrast cu fundalul, ea poate fi recunoscută foarte ușor de niște astfel de senzori.

Sistemul de afânare este rotit tot cu ajutorul unui servo motor sg90.

Schemă electrică:

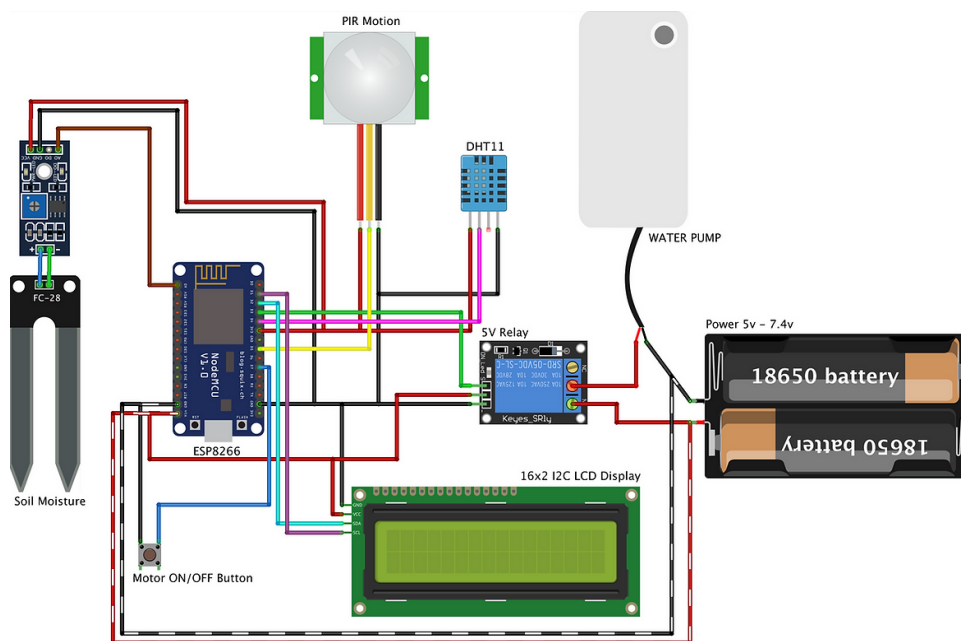


Sistemul de irigație și monitorizare a plantelor funcționează pe baza unei plăcuțe de dezvoltare esp8266. Aceasta pune la dispoziție numeroși pini pentru componente, cât și un modul wi-fi pentru conectarea la dispozitivul mobil.

Mai sunt folosiți niște senzori clasici pentru umiditatea solului (valori sub forma de procent), mișcare și un DHT11 pentru umiditatea și temperatura aerului (intervalul de temperatură de la 0°C la 50°C, iar umiditatea poate măsura de la 20% RH la 90% RH).

Pompa de apă este pornită și oprită de un modul releu acționat la nivel înalt.

Schemă electrică:



Complexitate

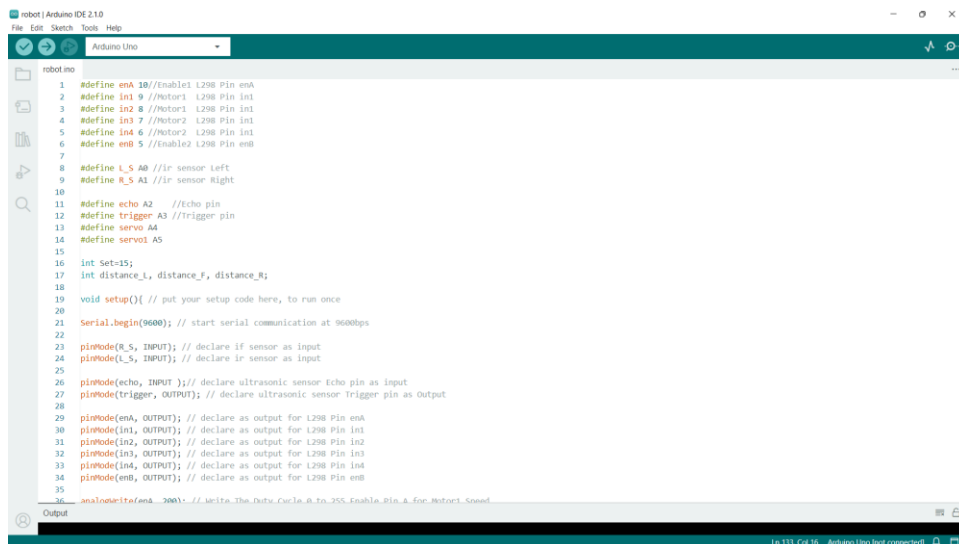
Tractorul nu este un simplu line follower, având capacitatea de a evita orice obstacol necunoscut.

Sistemul de irigație variază între unul telecomandat și unul automat.

- **Capitolul IV: Software**

Codul tractorului a fost realizat integral cu ajutorul aplicației Arduino, acesta fiind unul complet autonom.

În continuare vă voi prezenta anumite părți din acesta.



```
1 #define enA 10//enable1 L298 Pin enA
2 #define in1 9 //Motor1 L298 Pin in1
3 #define in2 8 //Motor1 L298 Pin in1
4 #define in3 7 //Motor2 L298 Pin in1
5 #define in4 6 //Motor2 L298 Pin in1
6 #define enB 5 //enable2 L298 Pin enB
7
8 #define L_S A0 //Ir sensor Left
9 #define R_S A1 //Ir sensor Right
10
11 #define echo A2 //Echo pin
12 #define trigger A3 //trigger pin
13 #define servo A4
14 #define servot A5
15
16 int Set=15;
17 int distance_L, distance_F, distance_R;
18
19 void setup(){ // put your setup code here, to run once
20
21   Serial.begin(9600); // start serial communication at 9600bps
22
23   pinMode(R_S, INPUT); // declare if sensor as input
24   pinMode(L_S, INPUT); // declare ir sensor as input
25
26   pinMode(echo, INPUT); // declare ultrasonic sensor Echo pin as input
27   pinMode(trigger, OUTPUT); // declare ultrasonic sensor Trigger pin as Output
28
29   pinMode(enA, OUTPUT); // declare as output for L298 Pin enA
30   pinMode(in1, OUTPUT); // declare as output for L298 Pin in1
31   pinMode(in2, OUTPUT); // declare as output for L298 Pin in2
32   pinMode(in3, OUTPUT); // declare as output for L298 Pin in3
33   pinMode(in4, OUTPUT); // declare as output for L298 Pin in4
34   pinMode(enB, OUTPUT); // declare as output for L298 Pin enB
35
36   analogWrite(enA, 200); // Write The Duty Cycle A To 255 Enable Pin A for Motor1 Speed
```

Definirea pinilor de pe plăcuță la care sunt conectate componentele pentru eficientizarea creării codului;



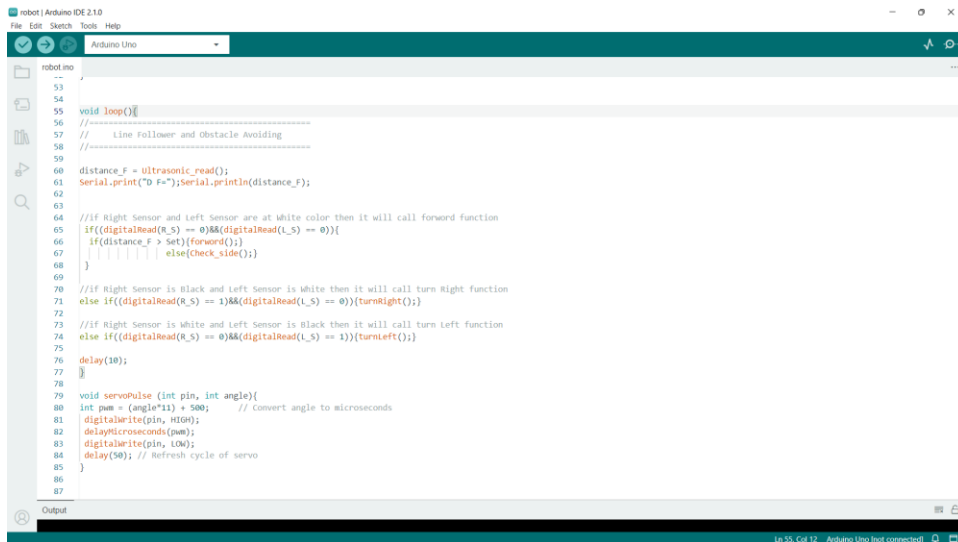
```
146 void forward() { //forward
147   digitalWrite(in1, LOW); //Left Motor backward Pin
148   digitalWrite(in2, HIGH); //Left Motor forward Pin
149   digitalWrite(in3, HIGH); //Right Motor forward Pin
150   digitalWrite(in4, LOW); //Right Motor backward Pin
151 }
152 }
153
154 void backward() { //backward
155   digitalWrite(in1, HIGH); //Left Motor backward Pin
156   digitalWrite(in2, LOW); //Left Motor forward Pin
157   digitalWrite(in3, LOW); //Right Motor forward Pin
158   digitalWrite(in4, HIGH); //Right Motor backward Pin
159 }
160
161 void turnRight() { //turnRight
162   digitalWrite(in1, LOW); //Left Motor backward Pin
163   digitalWrite(in2, HIGH); //Left Motor forward Pin
164   digitalWrite(in3, LOW); //Right Motor forward Pin
165   digitalWrite(in4, HIGH); //Right Motor backward Pin
166 }
167
168 void turnLeft() { //turnLeft
169   digitalWrite(in1, HIGH); //Left Motor backward Pin
170   digitalWrite(in2, LOW); //Left Motor forward Pin
171   digitalWrite(in3, HIGH); //Right Motor forward Pin
172   digitalWrite(in4, LOW); //Right Motor backward Pin
173 }
174
175 void stop() { //stop
176   digitalWrite(in1, LOW); //Left Motor backward Pin
177   digitalWrite(in2, LOW); //Left Motor forward Pin
178   digitalWrite(in3, LOW); //Right Motor forward Pin
179   digitalWrite(in4, LOW); //Right Motor backward Pin
180 }
```

Funcții care definesc sensul de rotație al motoarelor, în fiecare situație de mișcare.



```
119   delay(500);
120   turnLeft();
121   delay(500);
122 }
123 }
124
125 void check_side() {
126   Stop();
127   delay(100);
128   for (int angle = 70; angle <= 140; angle += 5) {
129     servoPulse(servo, angle);
130     delay(300);
131     distance_R = ultrasonic_read();
132     Serial.print("D R="); Serial.println(distance_R);
133     delay(100);
134     for (int angle = 140; angle >= 0; angle -= 5) {
135       servoPulse(servo, angle);
136       delay(500);
137       distance_L = ultrasonic_read();
138       Serial.print("D L="); Serial.println(distance_L);
139       delay(100);
140     }
141     for (int angle = 0; angle <= 70; angle += 5) {
142       servoPulse(servo, angle);
143       delay(300);
144       compareDistance();
145     }
146   }
147   digitalWrite(in1, LOW); //Left Motor backward Pin
148   digitalWrite(in2, HIGH); //Left Motor forward Pin
149   digitalWrite(in3, HIGH); //Right Motor forward Pin
150   digitalWrite(in4, LOW); //Right Motor backward Pin
151 }
152 }
153
154 void backward() { //backward
155   digitalWrite(in1, HIGH); //Left Motor backward Pin
156   digitalWrite(in2, LOW); //Left Motor forward Pin
157   digitalWrite(in3, LOW); //Right Motor forward Pin
158   digitalWrite(in4, HIGH); //Right Motor backward Pin
159 }
```

Funcție pentru ocolirea eficientă a obstacolelor.



```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87
// robot.ino
// Line Follower and Obstacle Avoiding
// Ultrasonic Sensor
// Digital Sensor
// Servo Motor

void loop()
{
    // Line Follower and Obstacle Avoiding
    // Ultrasonic Sensor
    distance_F = ultrasonic_read();
    Serial.print("D F=");Serial.println(distance_F);

    //if Right Sensor and Left Sensor are at white color then it will call forward function
    if((digitalRead(R_5) == 0)&&(digitalRead(L_5) == 0)){
        if(distance_F > Set){forward();}
        else{check_side();}
    }

    //if Right Sensor is Black and Left Sensor is White then it will call turn Right function
    else if((digitalRead(R_5) == 1)&&(digitalRead(L_5) == 0)){turnRight();}

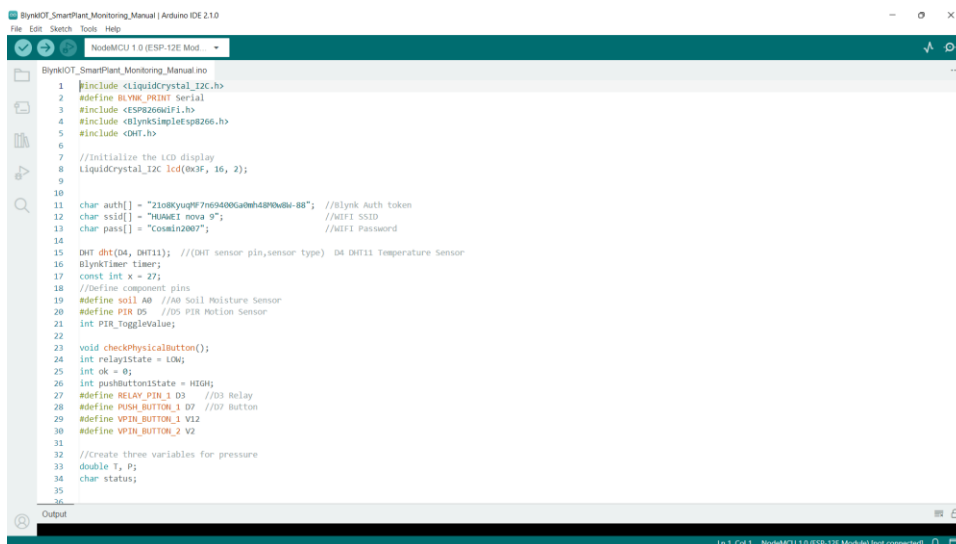
    //if Right Sensor is White and Left Sensor is Black then it will call turn Left function
    else if((digitalRead(R_5) == 0)&&(digitalRead(L_5) == 1)){turnLeft();}

    delay(10);
}

void servoPulse (int pin, int angle){
    int pwm = (angle*11) + 500; // Convert angle to microseconds
    digitalWrite(pin, HIGH);
    delayMicroseconds(pwm);
    digitalWrite(pin, LOW);
    delay(50); // Refresh cycle of servo
}
```

Funcția loop, care controlează urmarirea marcatului de pe jos.

Codul sistemului de irigație a fost realizat, de asemenea, cu ajutorul Arduino, împreună cu platforma Blynk, folosita pentru crearea aplicației mobile.



```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
BlynkIoT_SmartPlant_Monitoring_Manual.ino
#include <LiquidCrystal_I2C.h>
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>

//Initialize the LCD display
LiquidCrystal_I2C lcd(0x3F, 16, 2);

char auth[] = "21a8kyuqP7n04A0Gatnh09Nwbi-88"; //Blynk Auth token
char ssid[] = "HUMNET nova 9"; //WiFi SSID
char pass[] = "Cosmin2007"; //WiFi Password

DHT dht(D4, DHT11); //DHT sensor pin,sensor type) D4 DHT11 Temperature Sensor
BlynkTimer timer;
const int x = 27;

//Define component pins
#define soil_A0 //A0 Soil Moisture Sensor
#define PIR_D5 //D5 PIR Motion Sensor
int PIR_ToggleValue;

void checkPhysicalButton()
{
    int relayState = LOW;
    int ok = 0;
    int pushButtonState = HIGH;
    #define RELAY_PIN_1 D5 //D5 relay
    #define PUSH_BUTTON_1 D7 //D7 Button
    #define VPIN_BUTTON_1 V12
    #define VPIN_BUTTON_2 V2
    //Create three variables for pressure
    double T, P;
    char status;
}
```

Definirea pinilor de pe plăcuță la care sunt conectate componentele și conectarea la internet.


```
BlynkIoT_SmartPlant_Monitoring_Manual.ino | Arduino IDE 2.1.0
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12E Mod...)

BlynkIoT_SmartPlant_Monitoring_Manual.ino
66 timer.setInterval(5000, checkPhysicalButton);
67 }
68
69 //Get the DHT11 sensor values
70 void DHT11sensor() {
71   float h = dht.readHumidity();
72   float t = dht.readTemperature();
73
74   if (isnan(h) || isnan(t)) {
75     Serial.println("Failed to read from DHT sensor!");
76     return;
77   }
78   Blynk.virtualWrite(V0, t);
79   Blynk.virtualWrite(V1, h);
80
81   lcd.setCursor(0, 0);
82   lcd.print("T:");
83   lcd.print(t);
84
85   lcd.setCursor(8, 0);
86   lcd.print("H:");
87   lcd.print(h);
88 }
89
90 //Get the soil moisture values
91 void soilMoistureSensor() {
92   int value = analogRead(soil);
93   value = map(value, 0, 1024, 0, 100);
94   value = (value - 100) * -1;
95   Blynk.virtualWrite(V3, value);
96   if (ok) {
97     if (value < x) {
98       relayState = HIGH;
99       Blynk.virtualWrite(VPIN_BUTTON_1, relayState);
100       digitalWrite(RELAY_PIN_1, relayState);
101     } else {
102       relayState = LOW;
103       Blynk.virtualWrite(VPIN_BUTTON_1, relayState);
104       digitalWrite(RELAY_PIN_1, relayState);
105     }
106   } else {
107     if (value < x) {
108       Blynk.logEvent("sol_uscat", "Alertă! Udați plantele!");
109       lcd.setCursor(0, 1);
110       lcd.print("S:");
111       lcd.print(value);
112       lcd.print(" ");
113     }
114   }
115
116 //Get the PIR sensor values
117 void PIRsensor() {
118   bool value = digitalRead(PIR);
119   if (value) {
120     Blynk.logEvent("pir", "Alertă! Posibil animal prin apropiere"); //Enter your Event Name
121     digitalWrite(LED_V5);
122     LED.on();
123   }
124 }
125
126 BLYNK_WRITE(V6) {
127   PIR_ToggleValue = param.asInt();
128 }
129
130 BLYNK_CONNECTED() {
131   // Request the latest state from the server
132   Blynk.syncVirtual(VPIN_BUTTON_1);
133 }
```

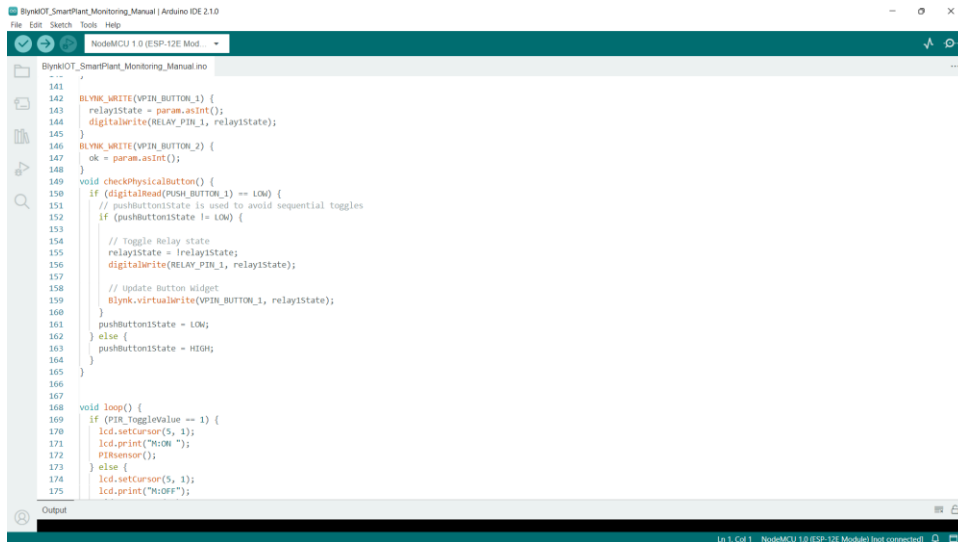
```
BlynkIoT_SmartPlant_Monitoring_Manual.ino | Arduino IDE 2.1.0
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12E Mod...)

BlynkIoT_SmartPlant_Monitoring_Manual.ino
91
92 //Get the soil moisture values
93 void soilMoistureSensor() {
94   int value = analogRead(soil);
95   value = map(value, 0, 1024, 0, 100);
96   value = (value - 100) * -1;
97   Blynk.virtualWrite(V3, value);
98   if (ok) {
99     if (value < x) {
100       relayState = HIGH;
101       Blynk.virtualWrite(VPIN_BUTTON_1, relayState);
102       digitalWrite(RELAY_PIN_1, relayState);
103     } else {
104       relayState = LOW;
105       Blynk.virtualWrite(VPIN_BUTTON_1, relayState);
106       digitalWrite(RELAY_PIN_1, relayState);
107     }
108   } else {
109     if (value < x) {
110       Blynk.logEvent("sol_uscat", "Alertă! Udați plantele!");
111       lcd.setCursor(0, 1);
112       lcd.print("S:");
113       lcd.print(value);
114       lcd.print(" ");
115     }
116   }
117
118 //Get the PIR sensor values
119 void PIRsensor() {
120   bool value = digitalRead(PIR);
121   if (value) {
122     Blynk.logEvent("pir", "Alertă! Posibil animal prin apropiere"); //Enter your Event Name
123     digitalWrite(LED_V5);
124     LED.on();
125   }
126 }
127
128 BLYNK_WRITE(V6) {
129   PIR_ToggleValue = param.asInt();
130 }
131
132 BLYNK_CONNECTED() {
133   // Request the latest state from the server
134   Blynk.syncVirtual(VPIN_BUTTON_1);
135 }
```

```
BlynkIoT_SmartPlant_Monitoring_Manual.ino | Arduino IDE 2.1.0
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12E Mod...)

BlynkIoT_SmartPlant_Monitoring_Manual.ino
105 relayState = LOW;
106 Blynk.virtualWrite(VPIN_BUTTON_1, relayState);
107 digitalWrite(RELAY_PIN_1, relayState);
108 }
109 } else {
110   if (value < x) {
111     Blynk.logEvent("sol_uscat", "Alertă! Udați plantele!");
112   }
113   lcd.setCursor(0, 1);
114   lcd.print("S:");
115   lcd.print(value);
116   lcd.print(" ");
117 }
118
119 //Get the PIR sensor values
120 void PIRsensor() {
121   bool value = digitalRead(PIR);
122   if (value) {
123     Blynk.logEvent("pir", "Alertă! Posibil animal prin apropiere"); //Enter your Event Name
124     digitalWrite(LED_V5);
125     LED.on();
126   } else {
127     digitalWrite(LED_V5);
128     LED.off();
129   }
130 }
131
132 BLYNK_WRITE(V6) {
133   PIR_ToggleValue = param.asInt();
134 }
135
136 BLYNK_CONNECTED() {
137   // Request the latest state from the server
138   Blynk.syncVirtual(VPIN_BUTTON_1);
139 }
```

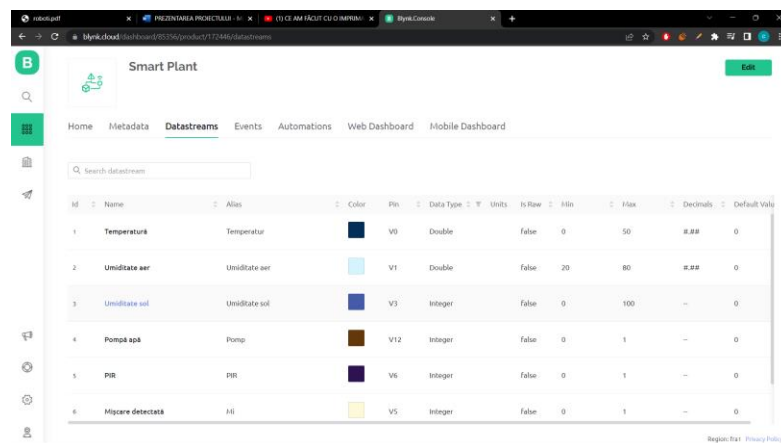
Funcții pentru citirea și afișarea valorilor de pe senzori.



```
141
142 BLYNK_WRITE(VPIN_BUTTON_1) {
143   relayState = param.asInt();
144   digitalWrite(RELAY_PIN_1, relayState);
145 }
146 BLYNK_WRITE(VPIN_BUTTON_2) {
147   ok = param.asInt();
148 }
149 void checkPhysicalButton() {
150   if (digitalRead(PUSH_BUTTON_1) == LOW) {
151     // pushbuttonState is used to avoid sequential toggles
152     if (pushButtonState != LOW) {
153       // Toggle Relay state
154       relayState = !relayState;
155       digitalWrite(RELAY_PIN_1, relayState);
156       // update button widget
157       Blynk.virtualWrite(VPIN_BUTTON_1, relayState);
158     }
159     pushButtonState = LOW;
160   } else {
161     pushButtonState = HIGH;
162   }
163 }
164
165
166
167
168 void loop() {
169   if (PIR_ToggleValue == 1) {
170     lcd.setCursor(5, 1);
171     lcd.print("Hi:ON ");
172     PIRsensor();
173   } else {
174     lcd.setCursor(5, 1);
175     lcd.print("Hi:OFF");
176   }
177 }
```

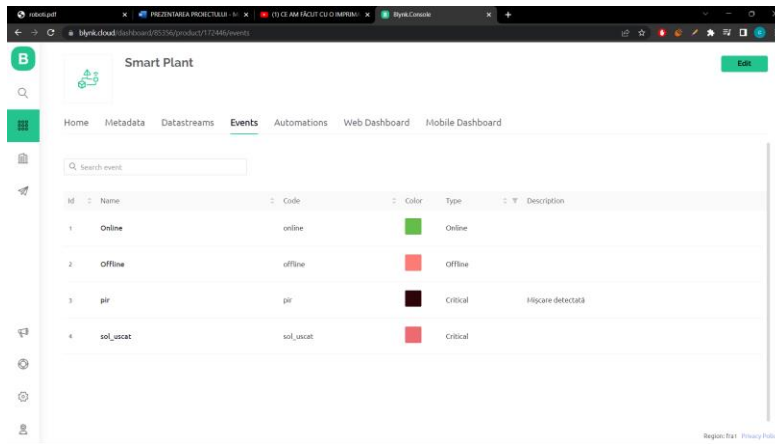
Funcție pentru utilizarea butonului fizic folosit la deschiderea și închiderea pompei.

Codul include și implementarea unui ecran LCD care să afișeze valorile senzorilor, însă modulul necesar pentru conectarea sa nu a ajuns la timp.



ID	Name	Alias	Color	Pin	Data Type	Units	Is Row	Min	Max	Decimals	Default Value
1	Temperatură	Temperatur	Dark Blue	V0	Double		false	0	50	2.##	0
2	Umiditate aer	Umiditate aer	Light Blue	V1	Double		false	20	80	2.##	0
3	Umiditate sol	Umiditate sol	Dark Blue	V3	Integer		false	0	100	--	0
4	Pompa apă	Pompa	Dark Brown	V12	Integer		false	0	1	--	0
5	PIR	PIR	Dark Purple	V6	Integer		false	0	1	--	0
6	Miscare detectată	Miscare	Yellow	V5	Integer		false	0	1	--	0

Pinii virtuali creati pe platforma Blynk



The screenshot shows a web application titled "Smart Plant" with a navigation bar containing "Home", "Metadata", "Datastreams", "Events", "Automations", "Web Dashboard", and "Mobile Dashboard". The "Events" tab is selected. Below the navigation bar is a search bar labeled "Search event:". A table displays the following data:

ID	Name	Code	Color	Type	Description
1	Online	online	Green	Online	
2	Offline	offline	Red	Offline	
3	pir	pir	Black	Critical	Înlocuire detectată
4	sol_uscat	sol_uscat	Red	Critical	

At the bottom right of the page, there are links for "Register" and "Privacy Policy".

Evenimentele folosite pentru notificări

- **Capitolul V: Design industrial**

Proiectul are posibilitate de scalare, modul de funcționare nefiind afectat în momentul utilizării unor componente mai mari.