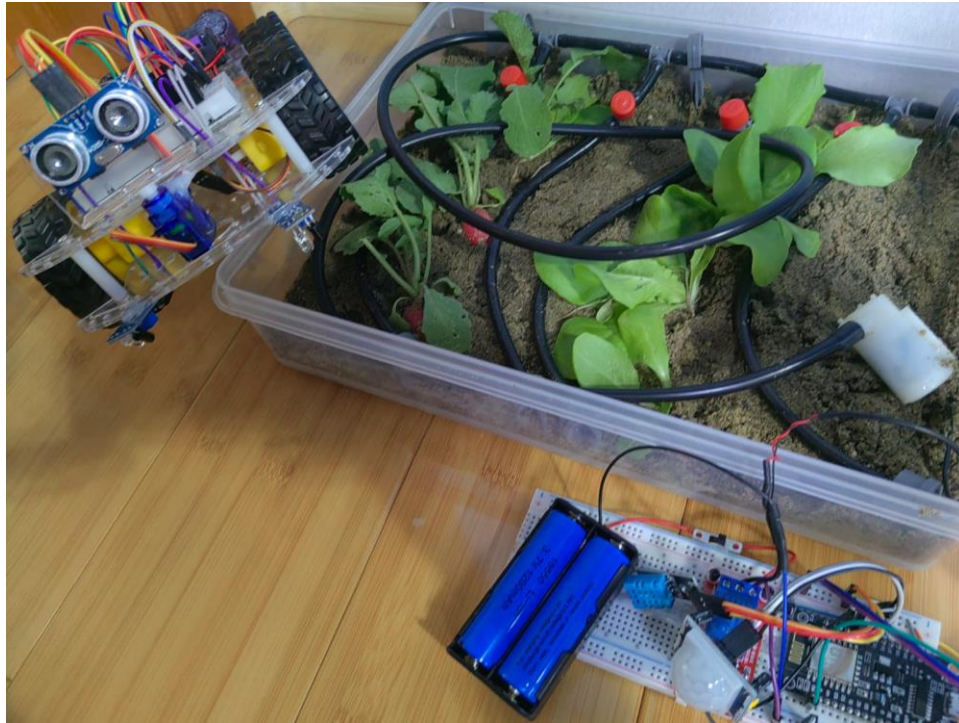


PREZENTAREA PROIECTULUI

„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. Modul automat oferă, de asemenea un sistem de temporizare, astfel încât udatul să fie realizat doar la orele potrivite.

„Cel mai rău moment pentru a uda plantele este la prânz. Orele în care există mai multă expunere la soare nu sunt bune pentru a furniza nutrienți plantelor noastre. Din cauza razelor puternice ale soarelui, apa se evaporă mai repede, ceea ce înseamnă că planta nu primește toată hrana de care are nevoie pentru a rămâne sănătoasă și puternică.

Dacă udăm plantele în plin soare, le putem deteriora grav din cauza schimbării termice pe care apa rece ar produce-o în contact cu planta caldă, datorită expunerii la soare”.

Sursa: www.impact.ro

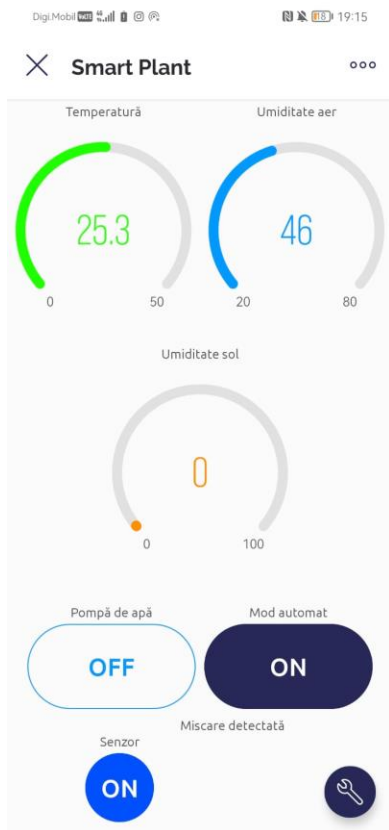
Sistemul este setat să pornească irigația între orele 9 seara și 9 dimineața.

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.

Este integrat un afișaj fizic LCD pentru monitorizarea plantelor în lipsa telefonului mobil.

De asemenea, proiectul dispune de un sistem de colectare al apei de ploaie. Unul dintre rezervoare include un senzor de nivel de apă. Când apa este de ajuns, este pornită pompa 1. Când nu există apă de ploaie în bazin, este pornită pompa 2, care simulează o fântână.

Colectarea apei de ploaie contribuie la reducerea costurilor, face sistemul să nu depindă de o singură sursă de apă, care ar putea avea o pană și, de asemenea, este benefică pentru mediu.



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**

Sisemul funcționează pe baza a 8 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 ultimele 2 se află în interiorul pompelor.

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, în cadrul sistemului de irigație, sunt alimentați constant pe timpul zilei de la un panou solar.

Încărcarea solară face sistemul în totalitate independent de alte surse de energie, acumulatorii încărcându-se în medie 6-7 ore pe zi.

Încărcarea se realizează ziua, iar consumul de energie este făcut în mare parte pe timpul nopții, pompele pornind doar în intervalul orar 21:00-9:00, energia acumulată pe timpul zilei fiind suficientă pentru a îl sustine.

- **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).

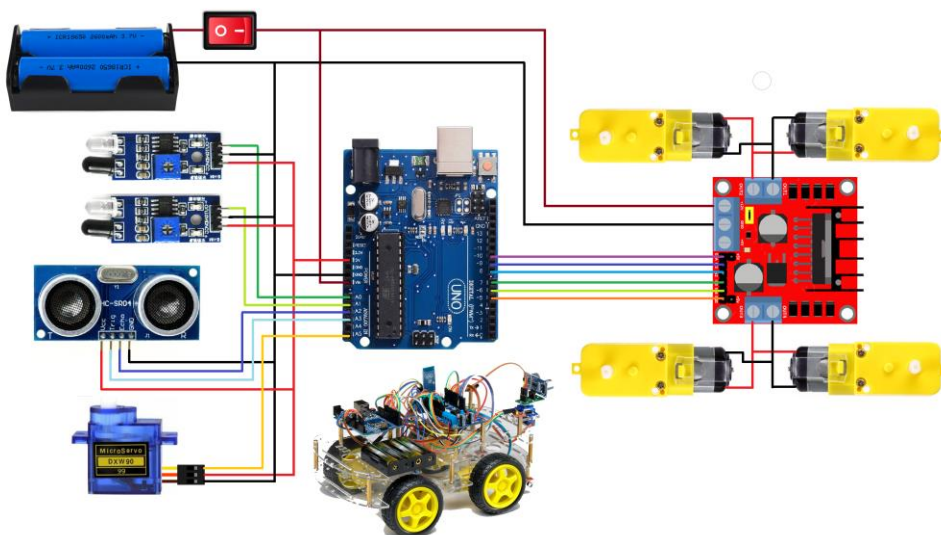
Motoarele sunt controlate de un driver de tip L298N, conectat la plăcuța Arduino, ele fiind eficient conectate între ele două câte două. Acest driver putea fi înlocuit cu un shield de motoare pentru Arduino, însă astfel s-ar fi limitat numărul de pini disponibili pentru restul componentelor.

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.

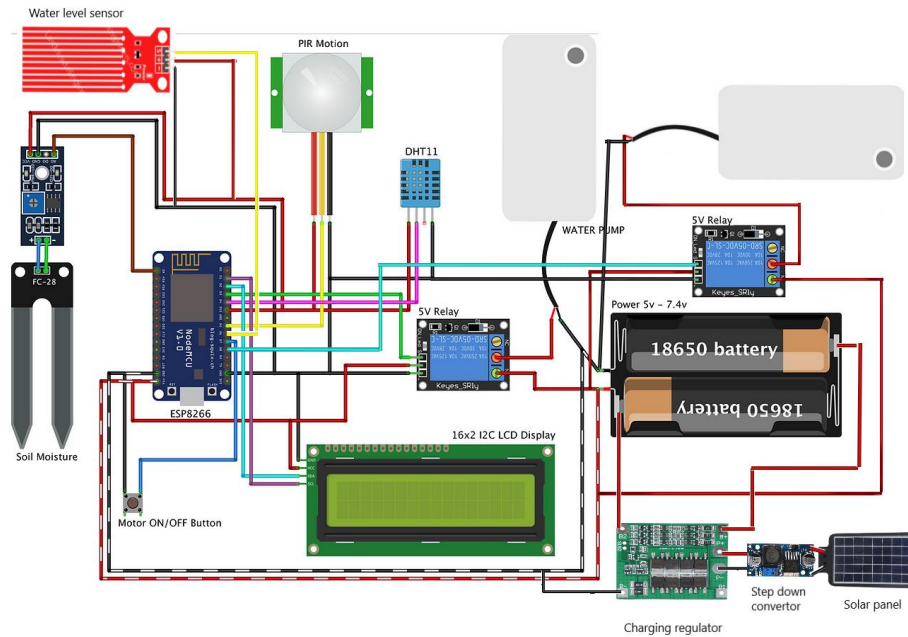
Mai sunt folosiți niște senzori clasici pentru umiditatea solului (valori sub forma de procent), mișcare și un DHT11 pentru umiditatea și temperatura aerului (intervalul de temperatură de la 0°C la 50°C, iar umiditatea poate măsura de la 20% RH la 90% RH).

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

Senzorul de apă comută între cele două pompe în cazul în care apa de ploaie din rezervor se termină.

Acumulatorii sunt încărcati de la un panou solar de 12v prin intermediul unui regulator de încărcare. Pentru protecția regulatorului este utilizat un step down convertor care aduce tensiunea produsă de panou (care variază între 9 și 15v) la o tensiune constantă de 8,4v. Acumulatorii sunt încărcăți individual (unul la 4,2v, iar cel de-al doilea, înseriat, la 8,4v).

Schemă electrică:



Complexitate

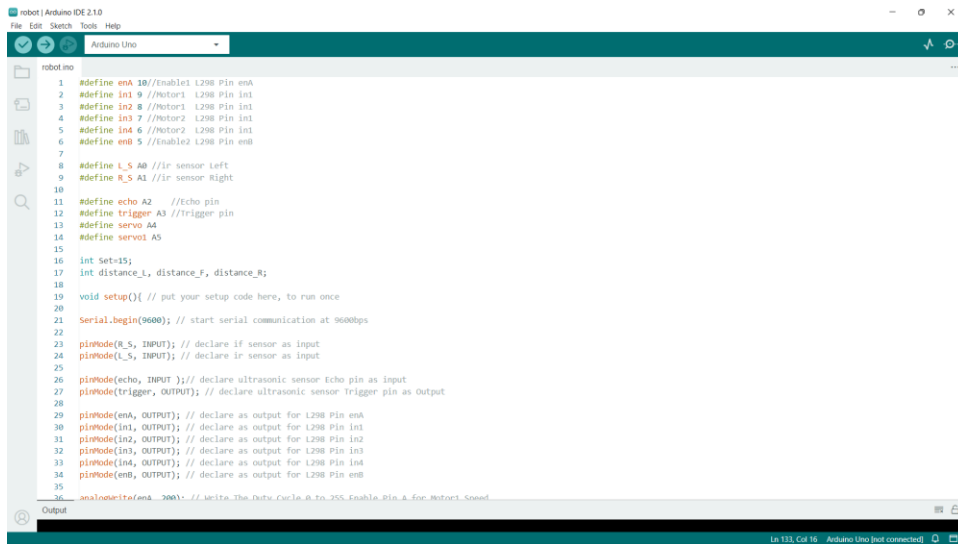
Tractorul nu este un simplu line follower, având capacitatea de a evita orice obstacol necunoscut.

Sistemul de irigație variază între unul telecomandat și unul automat.

- **Capitolul IV: Software**

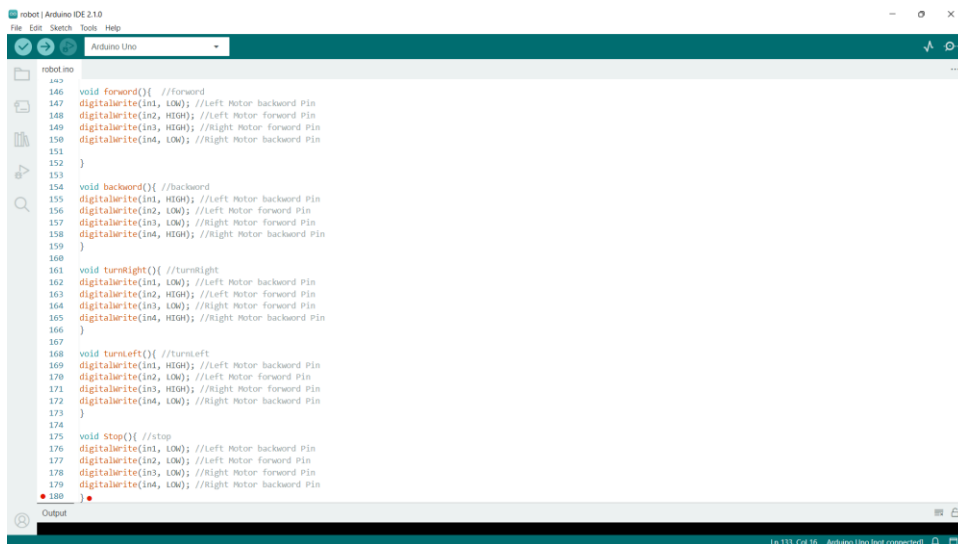
Codul tractorului a fost realizat integral cu ajutorul aplicației Arduino, acesta fiind unul complet autonom.

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 servo A5
15
16 int set=1;
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 Motor's Speed
```

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



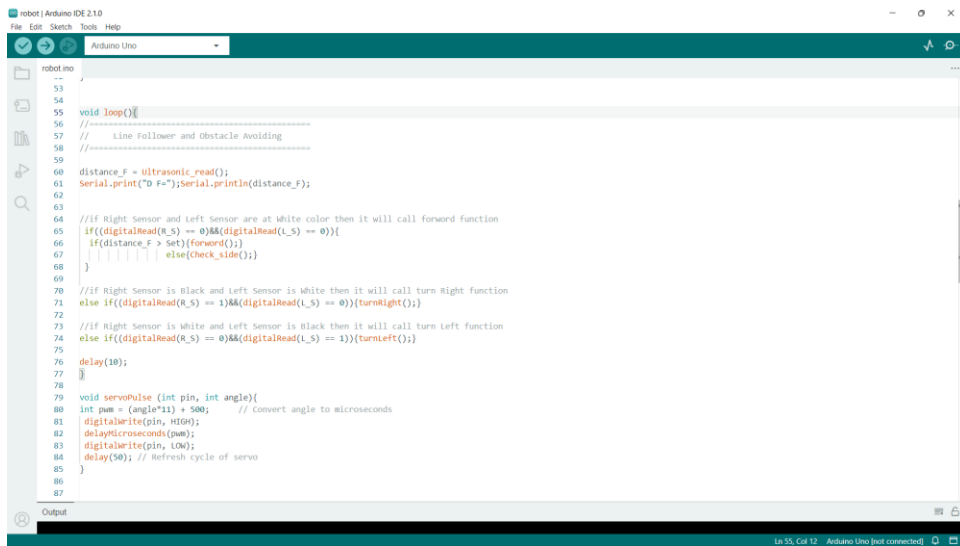
```
146 void forward(){ //forward
147   digitalWrite(in1, LOW); //Left Motor backward Pin
148   digitalWrite(in2, HIGH); //Left Motor forward Pin
149   digitalWrite(in3, HIGH); //Right Motor forward Pin
150   digitalWrite(in4, LOW); //Right Motor backward Pin
151 }
152
153
154 void backward(){ //backward
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 }
```

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



```
119 delay(600);
120 turnLeft();
121 delay(400);
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(100);
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   for (int angle = 0; angle <= 70; angle += 5) {
141     servoPulse(servo, angle); }
142   delay(300);
143   compareDistance();
144 }
145
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 }
```

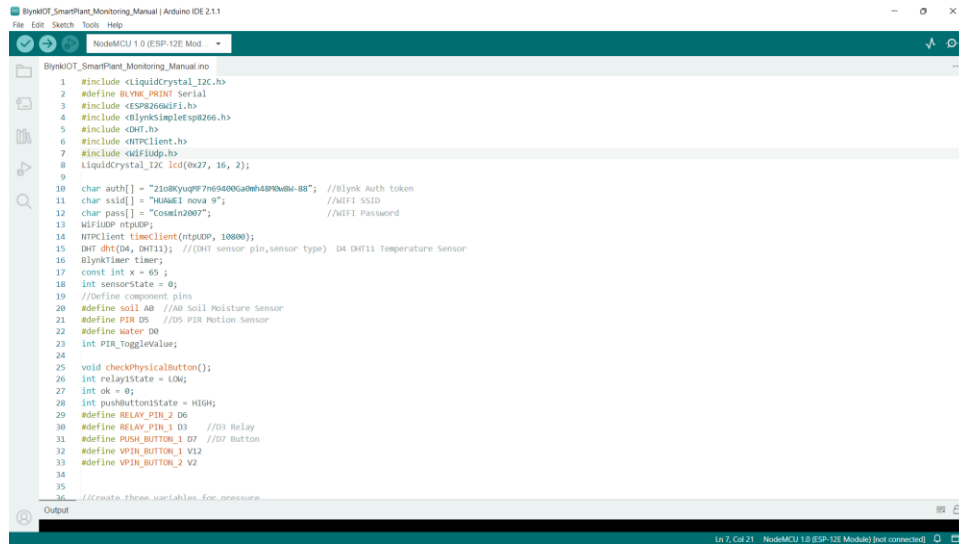
Funcție pentru ocolirea eficientă a obstacolelor.



```
53
54
55 void loop(){
56   //=====
57   // Line Follower and Obstacle Avoiding
58   //=====
59
60   distance_F = Ultrasonic_read();
61   Serial.print("D F=");Serial.println(distance_F);
62
63
64   //if Right Sensor and Left Sensor are at White color then it will call forward function
65   if((digitalRead(R_S) == 0)&&(digitalRead(L_S) == 0)){
66     if(distance_F > 50){forward();}
67     else{check_side();}
68   }
69
70   //if Right Sensor is Black and Left Sensor is White then it will call turn Right function
71   else if((digitalRead(R_S) == 1)&&(digitalRead(L_S) == 0)){turnRight();}
72
73   //if Right Sensor is White and Left Sensor is Black then it will call turn Left function
74   else if((digitalRead(R_S) == 0)&&(digitalRead(L_S) == 1)){turnLeft();}
75
76   delay(10);
77 }
78
79 void servoPulse (int pin, int angle){
80   int pwm = (angle*11) + 500; // Convert angle to microseconds
81   digitalWrite(pin, HIGH);
82   delayMicroseconds(pwm);
83   digitalWrite(pin, LOW);
84   delay(50); // Refresh cycle of servo
85 }
86
87
```

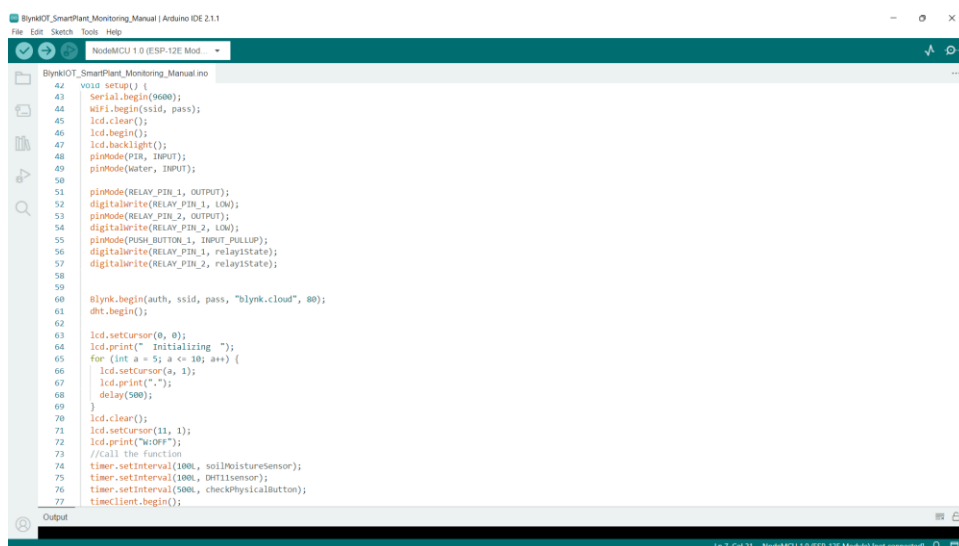
Funcția loop, care controlează urmarirea marcajului de pe jos.

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



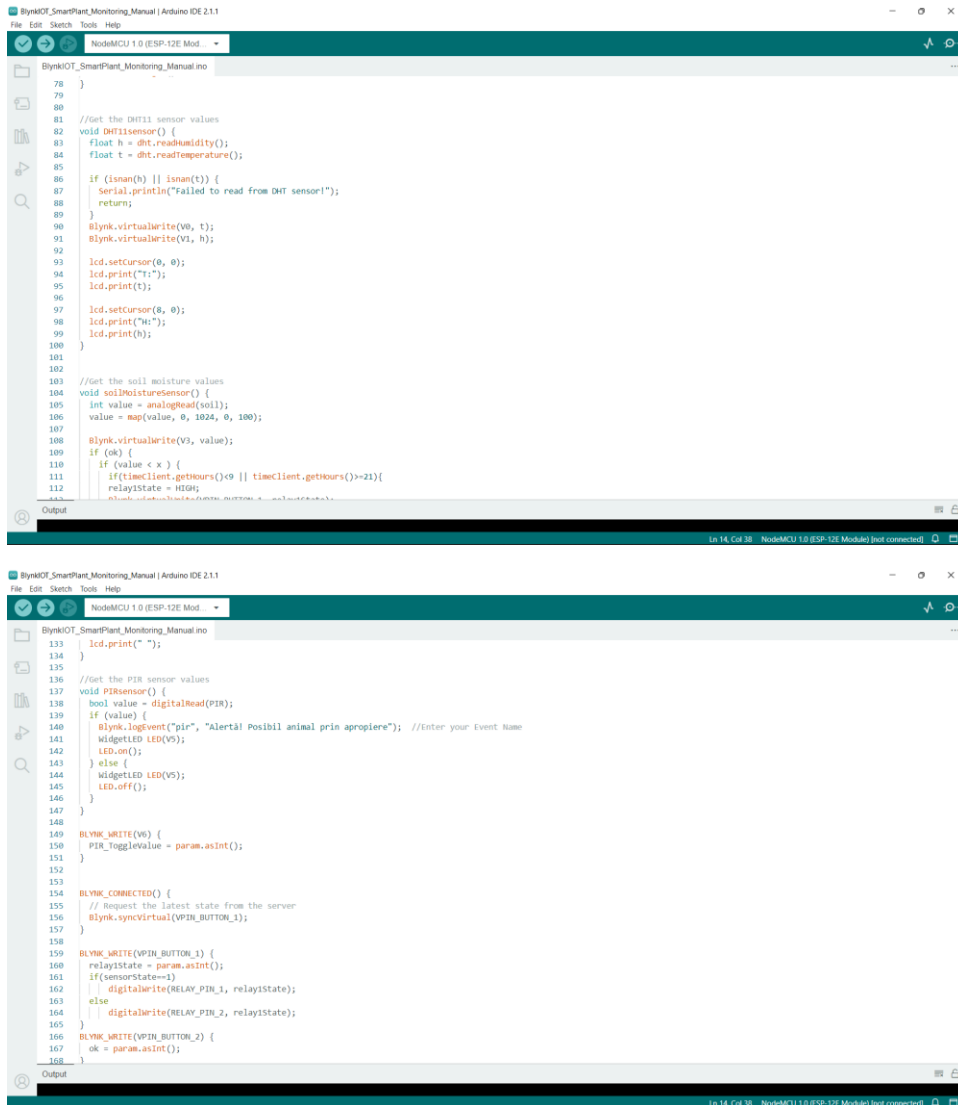
```
1 #include <LiquidCrystal_T2C.h>
2 #define BLYNK_PRINT Serial
3 #include <ESP8266WiFi.h>
4 #include <blynksimpleesp266.h>
5 #include <Dht.h>
6 #include <NTPClient.h>
7 #include <DHT11.h>
8 LiquidCrystal_T2C lcd(0x27, 16, 2);
9
10 char auth[] = "2108kyug7e7n6400Ga0mh20Nw0u-88"; //Blynk Auth token
11 char ssid[] = "NAA001 nova 9"; //WiFi SSID
12 char pass[] = "Cosmin2007"; //WiFi Password
13 WiFiUDP ntpUDP;
14 NTPClient timeClient(ntpUDP, 10000);
15 DHT dht(D4, DHT11); //DHT sensor pin,sensor type) D4 DHT11 Temperature Sensor
16 BlynkTimer timer;
17 const int x = 65;
18 int sensorState = 0;
19 //define component pins
20 #define soil_A0 //A0 Soil Moisture Sensor
21 #define PIR_D5 //D5 PIR Motion Sensor
22 #define Water_D0
23 int PIR_ToggleValue;
24
25 void checkPhysicalButton();
26 int relayState = LOW;
27 int ok = 0;
28 int pushButtonState = HIGH;
29 #define RELAY_PIN_2 D6
30 #define RELAY_PIN_1 D3 //D3 relay
31 #define PUSH_BUTTON_1 D7 //D7 button
32 #define VPIN_BUTTON_1 V12
33 #define VPIN_BUTTON_2 V2
34
35 //create these variables for measure
```

Definirea pinilor de pe plăcuță la care sunt conectate componentele și conectarea la serverul NTP pentru a avea acces la ora exactă.



```
42 void setup() {
43   Serial.begin(9600);
44   WiFi.begin(ssid, pass);
45   lcd.clear();
46   lcd.begin();
47   lcd.backlight();
48   pinMode(PIR, INPUT);
49   pinMode(Water, INPUT);
50
51   pinMode(RELAY_PIN_1, OUTPUT);
52   digitalWrite(RELAY_PIN_1, LOW);
53   pinMode(RELAY_PIN_2, OUTPUT);
54   digitalWrite(RELAY_PIN_2, LOW);
55   pinMode(PUSH_BUTTON_1, INPUT_PULLUP);
56   digitalWrite(RELAY_PIN_1, relayState);
57   digitalWrite(RELAY_PIN_2, relayState);
58
59
60   Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
61   dht.begin();
62
63   lcd.setCursor(0, 0);
64   lcd.print(" Initializing ");
65   for (int a = 5; a <= 100; a++) {
66     lcd.setCursor(a, 1);
67     lcd.print(".");
68     delay(500);
69   }
70   lcd.clear();
71   lcd.setCursor(11, 1);
72   lcd.print("Welcome");
73   //call the function
74   timer.setInterval(1000, soilMoistureSensor);
75   timer.setInterval(1000, DHT11sensor);
76   timer.setInterval(500, checkPhysicalButton);
77   timeClient.begin();
78 }
```

Funcția setup(inițializare rele, conectare la internet).



```
78 }
79
80 //Get the DHT11 sensor values
81 void DHT11sensor() {
82   float h = dht.readHumidity();
83   float t = dht.readTemperature();
84
85   if (isnan(h) || isnan(t)) {
86     Serial.println("Failed to read from DHT sensor!");
87     return;
88   }
89   Blynk.virtualWrite(V0, t);
90   Blynk.virtualWrite(V1, h);
91
92   lcd.setCursor(0, 0);
93   lcd.print("T:");
94   lcd.print(t);
95
96   lcd.setCursor(8, 0);
97   lcd.print("H:");
98   lcd.print(h);
99 }
100
101
102 //Get the soil moisture values
103 void soilMoisturesensor() {
104   int value = analogRead(soli);
105   value = map(value, 0, 1024, 0, 100);
106   Blynk.virtualWrite(V3, value);
107   if (ok) {
108     if (value < x) {
109       if (timeClient.getHours() > 9 || timeClient.getHours() > 21) {
110         relayState = HIGH;
111       }
112     }
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   } else {
124     digitalWrite(LED_V5);
125     LED.off();
126   }
127 }
128
129 BLYNK_WRITE(V6) {
130   PIR_ToggleValue = param.asInt();
131 }
132
133 BLYNK_CONNECTED() {
134   // Request the latest state from the server
135   Blynk.syncVirtual(VPIN_BUTTON_1);
136 }
137
138 BLYNK_WRITE(VPIN_BUTTON_1) {
139   relayState = param.asInt();
140   if (sensorState == 1) {
141     digitalWrite(RELAY_PIN_1, relayState);
142   } else {
143     digitalWrite(RELAY_PIN_2, relayState);
144   }
145 }
146 BLYNK_WRITE(VPIN_BUTTON_2) {
147   ok = param.asInt();
148 }
```

Funcții pentru citirea și afișarea valorilor de pe senzori.

```
159 BLYNK_WRITE(VPIN_BUTTON_1) {  
160   relayState = param.asInt();  
161   if(sensorState==1)  
162     digitalWrite(RELAY_PIN_1, relayState);  
163   else  
164     digitalWrite(RELAY_PIN_2, relayState);  
165 }  
166 BLYNK_WRITE(VPIN_BUTTON_2) {  
167   ok = param.asInt();  
168 }  
169 void checkPhysicalButton() {  
170   if (digitalRead(PUSH_BUTTON_1) == LOW) {  
171     // pushButtonState is used to avoid sequential toggles  
172     if (pushButtonState != LOW) {  
173       // Toggle Relay state  
174       relayState = !relayState;  
175       if(sensorState==1)  
176         digitalWrite(RELAY_PIN_1, relayState);  
177       else  
178         digitalWrite(RELAY_PIN_2, relayState);  
179       // update Button Widget  
180       Blynk.virtualWrite(VPIN_BUTTON_1, relayState);  
181     }  
182     pushButtonState = LOW;  
183   } else {  
184     pushButtonState = HIGH;  
185   }  
186 }  
187 }  
188 }  
189 }  
190 }  
191 void loop() {  
192   timeClient.update();  
193   sensorState = digitalRead(D0);  
194   if (PIR_ToggleValue == 1) {
```

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

```
100 }  
101 }  
102 }  
103 //Get the soil moisture values  
104 void soilMoistureSensor() {  
105   int value = analogRead(SOIL);  
106   value = map(value, 0, 1024, 0, 100);  
107 }  
108 Blynk.virtualWrite(V3, value);  
109 if (ok) {  
110   if (value < x) {  
111     if(timeClient.getHours()<9 || timeClient.getHours()>21){  
112       relayState = HIGH;  
113       Blynk.virtualWrite(VPIN_BUTTON_1, relayState);  
114       if(sensorState==1)  
115         digitalWrite(RELAY_PIN_1, relayState);  
116       else  
117         digitalWrite(RELAY_PIN_2, relayState);  
118     } else {  
119       relayState = LOW;  
120       Blynk.virtualWrite(VPIN_BUTTON_1, relayState);  
121       if(sensorState==1)  
122         digitalWrite(RELAY_PIN_1, relayState);  
123       else  
124         digitalWrite(RELAY_PIN_2, relayState);  
125     }  
126   } else {  
127     if (value < x)  
128       Blynk.logEvent("sol_uscat", "Alertă! Udați plantele");  
129   }  
130   lcd.setCursor(0, 1);  
131   lcd.print("S:");  
132   lcd.print(value);  
133   lcd.print(" ");  
134 }
```

Funcție care controlează acțiunile asociate umidității solului(modul automat, verificarea orei).

```

180
181 // update Button Widget
182 Blynk.virtualWrite(VPIN_BUTTON_1, relayState);
183 }
184 pushButtonState = LOW;
185 } else {
186   pushButtonState = HIGH;
187 }
188 }
189
190
191 void loop() {
192   timeClient.update();
193   sensorState = digitalRead(D0);
194   if (PIR_toggleLevel == 1) {
195     lcd.setCursor(5, 1);
196     lcd.print("Hi:ON ");
197     PIRsensor();
198   } else {
199     lcd.setCursor(5, 1);
200     lcd.print("Hi:OFF");
201     WidgetLED(LED(V5));
202     lcd.off();
203   }
204
205   if (relayState == HIGH) {
206     lcd.setCursor(11, 1);
207     lcd.print("Wi:ON ");
208   } else if (relayState == LOW) {
209     lcd.setCursor(11, 1);
210     lcd.print("Wi:OFF");
211   }
212   Blynk.run(); //Run the Blynk library
213   timer.run(); //Run the Blynk timer
214
215 }

```

Funcția loop (update oră, afișare lcd).

The screenshot shows the 'Datastreams' tab in the Blynk web interface. It displays a table of data streams for the 'Smart Plant' project. The table has columns for ID, Name, Alias, Color, Pin, Data Type, Units, Is Raw, Min, Max, Decimals, and Default Value.

ID	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Max	Decimals	Default Value
1	Temperatură	Temperatură	Dark Blue	V0	Double		false	0	50	2.00	0
2	Umiditate aer	Umiditate aer	Light Blue	V1	Double		false	20	80	2.00	0
3	Umiditate sol	Umiditate sol	Dark Blue	V3	Integer		false	0	100	—	0
4	Pompă apă	Pompă	Brown	V12	Integer		false	0	1	—	0
5	PIR	PIR	Dark Purple	V6	Integer		false	0	1	—	0
6	Miscare detectată	Miscare detectată	Yellow	V5	Integer		false	0	1	—	0

Pinii virtuali creati pe platforma Blynk

The screenshot shows the 'Events' tab in the Blynk web interface. It displays a table of events for the 'Smart Plant' project. The table has columns for ID, Name, Code, Color, Type, and Description.

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

Evenimentele folosite pentru notificări

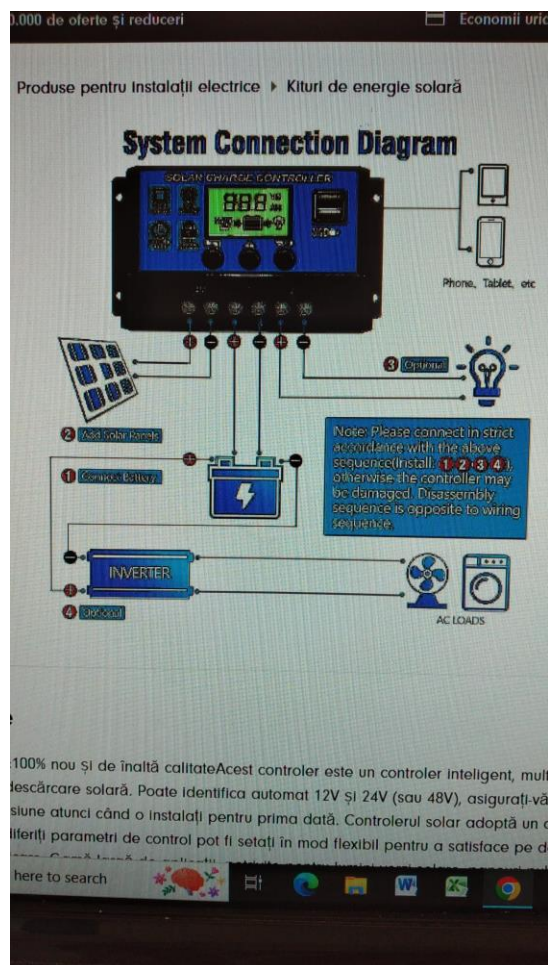
Codul este realizat astfel încât sistemul să fie cât mai intuitiv, iar

- **Capitolul V: Design industrial**

Proiectul are posibilitate de scalare. Pentru prezentare, în cadrul tractorului au fost folosiți senzori cu infraroșu pentru urmărirea traseului. La scară largă, aceștia vor fi înlocuiți cu niște senzori cu ultrasunete pentru urmărirea rândului arabil.

În cazul sistemului de irigație, la scara largă ar putea fi folosită o electrovalvă conectată la rezervorul de apă de ploaie pentru eficiență, în locul pompei. O electrovalvă este un robinet acționat electric.

Încărcarea s-ar realiza cu ajutorul unui panou solar mai mare, conectat la un controler de încărcare.



Va fi folosit si un invertor pentru a limita puterea extrasa din acumulatori si a nu descarca bateria necontrolabil.