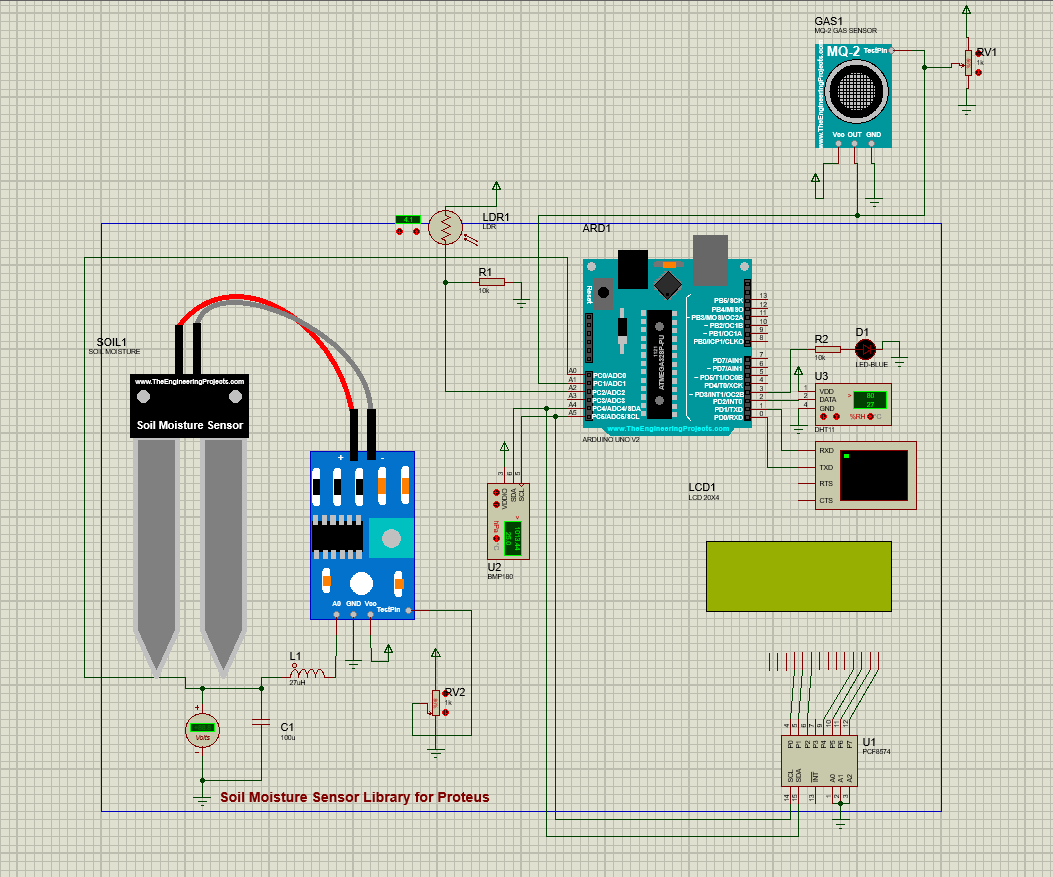
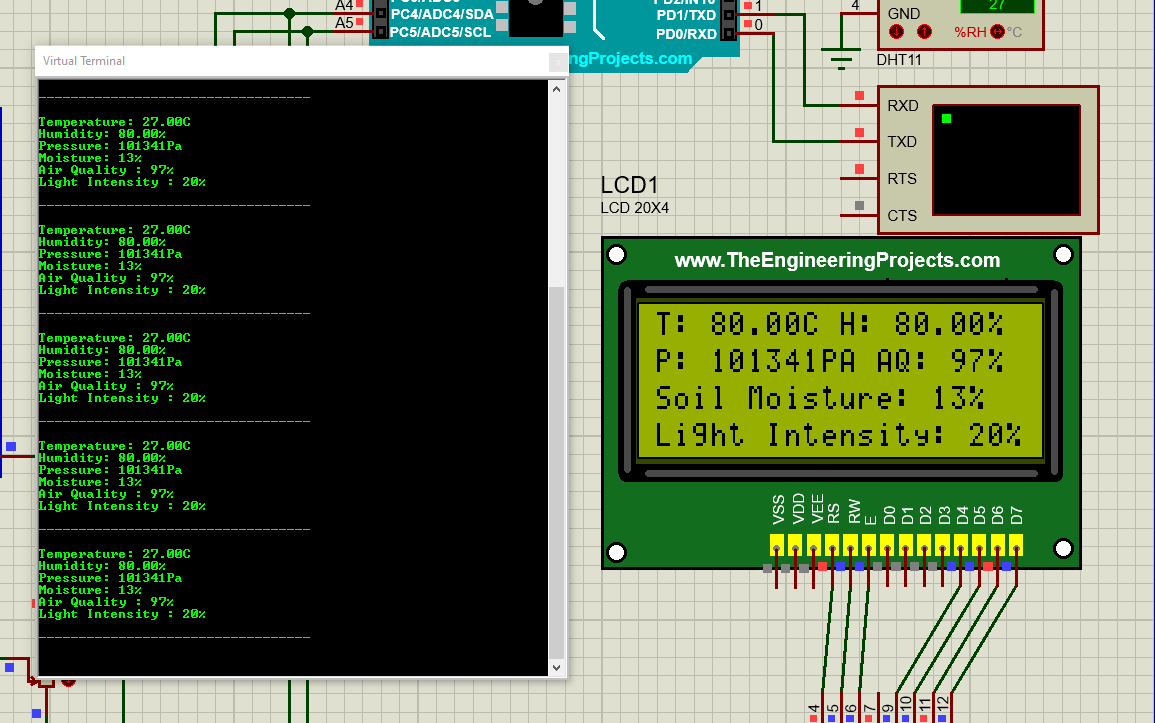
Pentru simularea proiectului vom folosi aplicatia Proteus, unde putem incerca codul Arduino si observa starea senzorilor inainte de a realiza schema fizica.

Pentru acest proiect am ales urmatoarele componente:

1. Arduino UNO R3
2. Senzor de umiditate rezistiv
3. Senzor de calitatea aerului MQ-2
4. Senzor de presiune
5. Senzor de intensitate luminoasa
6. Ecran LCD
7. Alte componente necesare testarii circuitului pecum potentiometer si leduri



Putem observa starea senzorilor setati de noi cu ajutorul unor potentiometer.



Am reusit sa obstinem aceste detalii cu ajutorul codului Arduino:

//Libraries

#include<Wire.h>

#include<Adafruit\_Sensor.h>

#include<DHT.h>

#include<DHT\_U.h>

#include<Adafruit\_BMP085.h>

#include<LiquidCrystal\_I2C.h>

//Pins declaration

#define DHTPIN 2

#define soil\_pin A0

#define gas\_pin A1

#define ldr\_pin A2

#define valve 3

#define DHTTYPE DHT11

//Sensor object

DHT\_Unified dht(DHTPIN, DHTTYPE);

Adafruit\_BMP085 bmp;

LiquidCrystal\_I2C lcd(0x20,20,4);

//Declare data variables

int soil\_value;

int gas\_value;

int ldr\_value;

uint32\_t dedlayMS;

int delayMS;

void setup(){

  Serial.begin(9600);

  Serial.println("Sistem inteligent de irigatii");

  pinMode(valve, OUTPUT);

  lcd.init();

  lcd.backlight();

  dht.begin();

  if(!bmp.begin()){

    Serial.println("Could not find a valid sensor");

    while(1){}

  }

  else{

    Serial.println(F("Pressure sensor initialized"));

  }

  Serial.println(F("DHT11 Begin"));

  Serial.println("");

  delayMS = sensor.min\_delay / 1000;

  lcd.clear();

}

void loop(){

  delay(delayMS);

  //Get temperature

  sensors\_event\_t event;

  dht.temperature().getEvent(&event);

  if (isnan(event.temperature)){

    Serial.println(F("Error reading temperature"));

  }

  else{

    Serial.print(F("Temperature: "));

    Serial.print(event.temperature);

    Serial.println(F("C"));

  }

  //Get humidity

  dht.humidity().getEvent(&event);

  if (isnan(event.relative\_humidity)){

    Serial.println(F("Error reading humidity"));

  }

  else{

    Serial.print(F("Humidity: "));

    Serial.print(event.relative\_humidity);

    Serial.println(F("%"));

  }

  //Get pressure

   Serial.print(F("Pressure: "));

    Serial.print(bmp.readPressure());

    Serial.println(F("Pa"));

  //Get Soil Value

  soil\_value = analogRead(soil\_pin);

  soil\_value = map(soil\_value,0,1023,0,100);

  Serial.print("Moisture: ");

  Serial.print(soil\_value);

  Serial.println("%");

  if(soil\_value<25){

    digitalWrite(valve,HIGH);

  }

  else{

    digitalWrite(valve,LOW);

  }

  //Gas Sensor

  gas\_value = analogRead(gas\_pin);

  gas\_value = map(gas\_value,0,1023,100,0);

  Serial.print("Air Quality : ");

  Serial.print(gas\_value);

  Serial.println("%");

  //LDR

  ldr\_value = analogRead(ldr\_pin);

  ldr\_value = map(ldr\_value, 0 ,1023, 0, 100);

  Serial.print("Light Intensity : ");

  Serial.print(ldr\_value);

  Serial.println("%");

  Serial.println();

  Serial.println(F("----------------------------------"));

  Serial.println();

  //LCD display

  lcd.setCursor(0,0);

  lcd.print("T: ");

  lcd.print(event.temperature);

  lcd.print("C H: ");

  lcd.print(event.relative\_humidity);

  lcd.print("%");

  lcd.setCursor(0,1);

  lcd.print("P: ");

  lcd.print(bmp.readPressure());

  lcd.print("PA AQ: ");

  lcd.print(gas\_value);

  lcd.print("%");

  lcd.setCursor(0,2);

  lcd.print("Soil Moisture: ");

  lcd.print(soil\_value);

  lcd.print("%");

  lcd.setCursor(0,3);

  lcd.print("Light Intensity: ");

  lcd.print(ldr\_value);

  lcd.print("%");

}

Codul precedent este foarte simplu, momentan, doar luam valorile date de senzori si le printam pe ecranul lcd, urmand ca sa continuam prin face niste leduri sa se aprinda cand valoarea data de sensor nu este intr-o limita sanatoasa plantei, valorile minime si maxime ale senzorilor vor fi luate de catre un sistem intelligent de clasificare a plantelor, acesta le va clasifica si va lua valorile date de noi intr-o baza de date.

Camera reprezinta cea mai complexa parte a proiectului nostru, vom dori sa facem camera sa poata fi folosita real time cu un model programabil ESP32-CAM.

Pentru modelul de recunoastere a plantelor momentan vom folosi limbajul de programare Python,in care ne ajutam de librariile facute special pentru recunoasterea imaginilor: pyTorch si Tranformers,

Pentru functionalitatea acestora trebuie sa antrenam modelul cu imagini cu plante pana la recunoasterea acestora automata. Codul pe care il avem momentan contine 3 fisiere:

* Un processor de imagini care are rolul de a prelucra imaginile:

1. import os
2. from PIL import Image
3. def preprocess\_images(base\_path, plant\_dataset, size):
4. for category, data\_types in plant\_dataset.items():
5. for data\_type, file\_list in data\_types.items():
6. input\_dir = os.path.join(base\_path, category, data\_type)
7. output\_dir = os.path.join(base\_path, category, f'preprocessed\_{data\_type}')
8. if not os.path.exists(output\_dir):
9. os.makedirs(output\_dir)
10. for file in file\_list:
11. img\_path = next((os.path.join(input\_dir, filename) for filename in os.listdir(input\_dir) if filename.lower() == file.lower()), None)
12. if img\_path is None:
13. print(f"Image not found: {file}")
14. continue
15. print(f"Processing image: {img\_path}")
16. img = Image.open(img\_path)
17. img = img.resize(size)
18. img.save(os.path.join(output\_dir, file), format='JPEG')
19. # Your Python dataset definition
20. plant\_dataset = {
21. "tomato": {
22. "train": ["tomato\_train\_1.jpg", "tomato\_train\_2.jpg"],
23. "val": ["tomato\_val\_1.jpg"]
24. },
25. "dragon\_fruit": {
26. "train": ["dragon\_fruit\_train\_1.jpg", "dragon\_fruit\_train\_2.jpg"],
27. "val": ["dragon\_fruit\_val\_1.jpg"]
28. },
29. "cactus": {
30. "train": ["cactus\_train\_1.jpg", "cactus\_train\_2.jpg"],
31. "val": ["cactus\_val\_1.jpg"]
32. }
33. }
34. # Provide the base path to your plant photos
35. base\_path = 'C:\\Users\\silvi\\Desktop\\ProiectSE\\Plante'
36. # Specify the size for resizing images
37. image\_size = (224, 224)
38. # Preprocess images using the defined dataset
39. preprocess\_images(base\_path, plant\_dataset, image\_size)
40. print("Preprocessing complete.")

* Un fisier cu baza de date si dictionarul :

1. import os
2. from PIL import Image
3. from torchvision import transforms
4. from torch.utils.data import Dataset, DataLoader
5. class PlantDataset(Dataset):
6. def \_\_init\_\_(self, root\_dir, dataset\_dict, transform=None):
7. self.root\_dir = root\_dir
8. self.dataset\_dict = dataset\_dict
9. self.transform = transform
10. self.data = []
11. self.labels = []
12. self.load\_data()
13. def load\_data(self):
14. for category, splits in self.dataset\_dict.items():
15. for split, file\_list in splits.items():
16. for file\_name in file\_list:
17. img\_path = os.path.join(self.root\_dir, category, split, file\_name)
18. label = category  # You might need to convert categories to numeric labels
19. self.data.append(img\_path)
20. self.labels.append(label)
21. def \_\_len\_\_(self):
22. return len(self.data)
23. def \_\_getitem\_\_(self, idx):
24. img\_path = self.data[idx]
25. label = self.labels[idx]
26. img = Image.open(img\_path).convert("RGB")
27. if self.transform:
28. img = self.transform(img)
29. return img, label
30. # Example usage:
31. root\_dir = 'C:\\Users\\silvi\\Desktop\\ProiectSE\\Plante'
32. plant\_dataset\_dict = {
33. 'tomato': {
34. 'train': ["tomato\_train\_1.jpg", "tomato\_train\_2.jpg"],
35. 'val': ["tomato\_val\_1.jpg"]
36. },
37. 'dragon\_fruit': {
38. 'train': ["dragon\_fruit\_train\_1.jpg", "dragon\_fruit\_train\_2.jpg"],
39. 'val': ["dragon\_fruit\_val\_1.jpg"]
40. },
41. 'cactus': {
42. 'train': ["cactus\_train\_1.jpg", "cactus\_train\_2.jpg"],
43. 'val': ["cactus\_val\_1.jpg"]
44. }
45. }
46. transform = transforms.Compose([
47. transforms.Resize((224, 224)),
48. transforms.ToTensor(),
49. ])
50. plant\_dataset = PlantDataset(root\_dir, plant\_dataset\_dict, transform=transform)
51. # Create DataLoader
52. batch\_size = 32
53. data\_loader = DataLoader(plant\_dataset, batch\_size=batch\_size, shuffle=True)
54. # Example of how to iterate through the data loader
55. for inputs, labels in data\_loader:
56. # Your training loop goes here
57. print(f"Batch Size: {len(labels)}")

* Si un fisier pentru antrenarea modelului:

1. import os
2. import torch
3. import torch.nn as nn
4. import torch.optim as optim
5. from torch.utils.data import DataLoader
6. from torchvision import transforms
7. from torchvision.models import resnet18
8. from PIL import Image
9. # Custom dataset class
10. class PlantDataset(torch.utils.data.Dataset):
11. def \_\_init\_\_(self, root\_dir, transform=None):
12. self.root\_dir = root\_dira
13. self.transform = transform
14. self.data = self.load\_data()
15. def load\_data(self):
16. # Implement your logic to load dataset paths here
17. # Example: return a list of (image\_path, label) tuples
18. data = [
19. (os.path.join(self.root\_dir, 'tomato', 'train', 'tomato\_train\_1.jpg'), 0),
20. (os.path.join(self.root\_dir, 'dragon\_fruit', 'train', 'dragon\_fruit\_train\_1.jpg'), 1),
21. (os.path.join(self.root\_dir, 'cactus', 'train', 'cactus\_train\_1.jpg'), 2)
22. ]
23. return data
24. def \_\_len\_\_(self):
25. return len(self.data)
26. def \_\_getitem\_\_(self, idx):
27. img\_path, label = self.data[idx]
28. img = Image.open(img\_path).convert("RGB")
29. if self.transform:
30. img = self.transform(img)
31. return img, label
32. # Define transformations
33. transform = transforms.Compose([
34. transforms.Resize((224, 224)),
35. transforms.ToTensor(),
36. ])
37. # Set paths
38. root\_dir = 'C:\\Users\\silvi\\Desktop\\ProiectSE\\Plante'
39. # Create dataset and dataloader
40. train\_dataset = PlantDataset(root\_dir, transform=transform)
41. train\_loader = DataLoader(train\_dataset, batch\_size=32, shuffle=True)
42. # Define a simple neural network model
43. model = resnet18(pretrained=False, num\_classes=3)  # Adjust num\_classes based on your dataset
44. # Define loss function and optimizer
45. criterion = nn.CrossEntropyLoss()
46. optimizer = optim.Adam(model.parameters(), lr=0.001)
47. # Training loop
48. num\_epochs = 5
49. for epoch in range(num\_epochs):
50. for inputs, labels in train\_loader:
51. optimizer.zero\_grad()
52. outputs = model(inputs)
53. loss = criterion(outputs, labels)
54. loss.backward()
55. optimizer.step()
56. print(f'Epoch {epoch+1}/{num\_epochs}, Loss: {loss.item()}')
57. # Debugging prints
58. for img\_path, label in train\_dataset.data:
59. print(f"Image Path: {img\_path}, Label: {label}")
60. print('Training complete.')

Pentru aceste modele, momentan, folosim doar 3 imagini pentru 3 tipuri de plante.