Cook Pro Measure

Tobescu Dalya-Alexandra

Grupa 331CB

Introducere

Proiectul constă în construirea unui cântar de bucătărie utilizând un microcontroller Arduino și diverse componente hardware. Scopul acestuia este să ofere o modalitate precisă de măsurare a greutății ingredientelor în grame, facilitând procesul de gătit și urmarea rețetelor în bucătărie. Acest cântar este folositor atunci când utilizatorul dorește să determine cantitatea exactă a ingredientelor utilizate în preparatele culinare. De exemplu, este util în situațiile în care utilizatorul este implicat într-un regim alimentar specific sau în activități precum antrenamentele la sală și are nevoie să își calculeze masa alimentară zilnică.

Descriere generală

Cântarul se bazează pe o celulă de încărcare cu o capacitate de 5 kg, care convertește presiunea exercitată de obiectele plasate pe platformă într-un semnal analog. Acest semnal este apoi transmis la modulul de citire a greutății HX711, care îl transformă într-un semnal digital ce poate fi citit de către microcontrollerul Arduino. Prin intermediul unui ecran LCD, greutatea este afișată cu precizie utilizatorului. Este integrat un buton pentru a reseta greutatea curentă la zero, permițând efectuarea de măsurători repetate cu ușurință. De asemenea, este integrat un buton pentru functia de PLATE, in aceasta functie putem sa cantarim ce dorim fara a fi luata in calcul si farfuria cu care masuram.

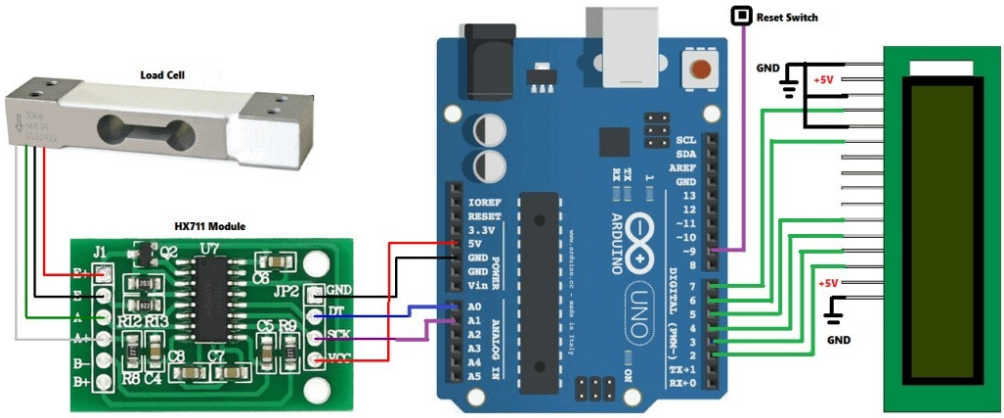
Hardware Design

Listă de piese necesare:

- Arduino
- Breadboard
- 1 LED
- Ecran LCD 16×2
- Potentiometru
- Rezistente

- Celulă de încărcare 5kg
- Modul citire greutate HX711
- 2 butoane
- Fire
- Suport greutati

Cum arata circuitul:



Schema EAGLE:

Conectare piese hardware:

Ecran LCD:

1. VSS pe GND
2. VDD pe 3.3V
3. V0 pe OUTPUT la potentiometru
4. RS pe pinul 13
5. RW pe GND
6. E pe pinul 12
7. D4 pe pinul 4
8. D5 pe pinul 5
9. D6 pe pinul 6
10. D7 pe pinul 7
11. A conectat la o rezistenta
12. K pe GND

Modulul HX711:

1. GND pe GND la arduino
2. DT pe pinul 2 arduino
3. SCK pe pinul 3 arduino
4. VCC la 5V

Legare modul HX711 si Load Cell:

1. E+ cu firul rosu al celulei
2. E- cu firul albastru al celulei
3. A- cu firul verde al celulei
4. A+ cu firul alb al celulei

Buton RESET:

- pe pinul 8 Arduino

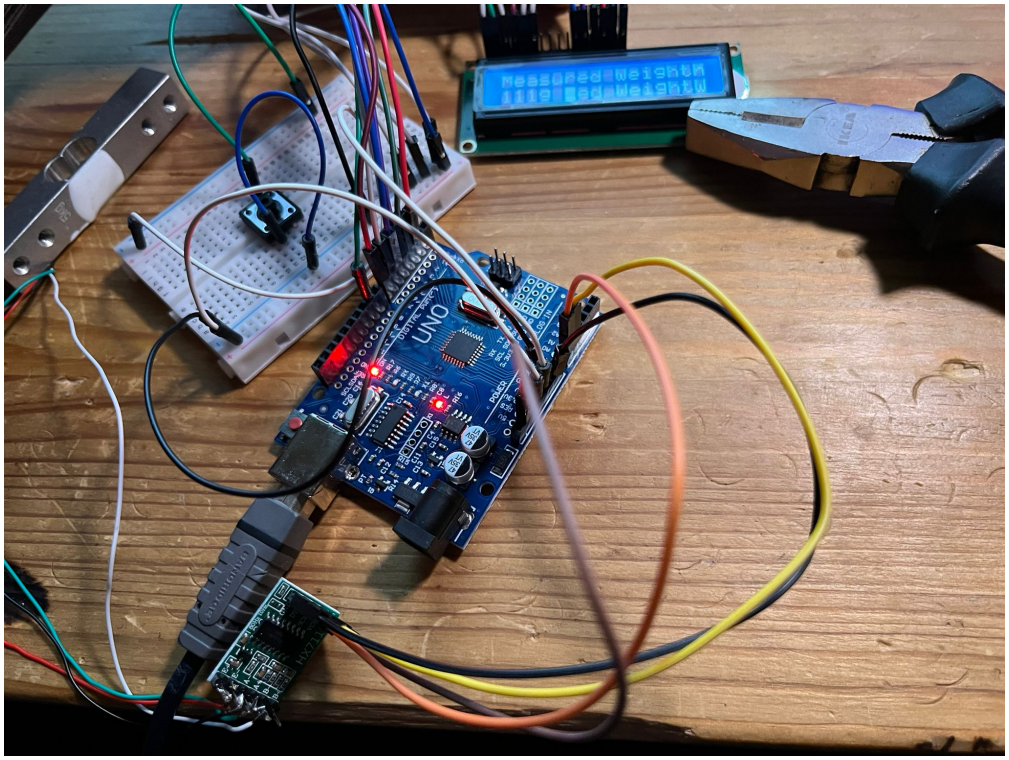
Buton PLATE FUNCTION:

- pe pinul 10 Arduino

LED:

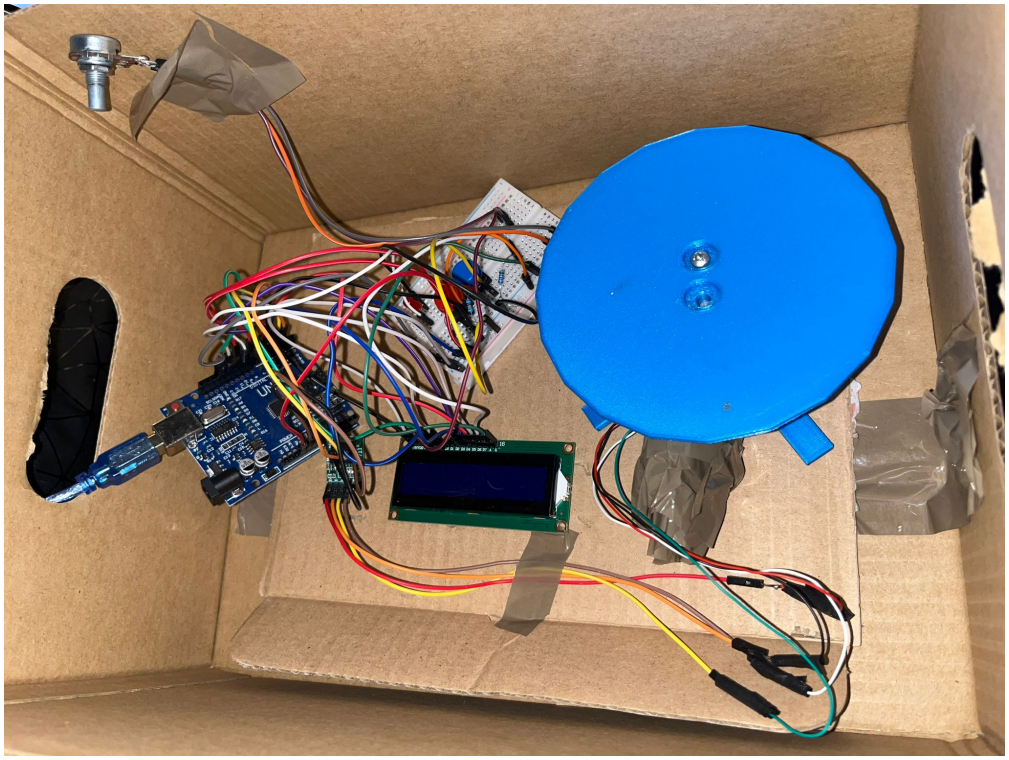
- pe pinul 9 Arduino

Implementarea hardware dupa schema electrica in faza initiala (fara software):



Dupa cum se poate observa am implementat circuitul dupa schema in Tinkercad de mai sus si am incercat sa implementez codul, insa momentan sunt cateva probleme. Firele de la ecranul LCD nu stau corespunzator si intalnesc eroare de tip Black Box, astfel, daca nu tin ecranul intr-o pozitie anume, ecranul afișează doar pătrate negre (sau “black boxes”) în loc de text sau informațiile așteptate. Probabil, o sa rezolv problema cu ajutorul unui potentiometru sau o sa incerc sa folosesc alte fire.

Implementare hardware in faza finala:



Am rezolvat problema cu Black Boxes, adaugand un potentiometru cu ajutorul caruia setez luminozitatea ecranului. Am adaugat led-ul care imi indica cand cantarul este in Plate Function si un buton cu care intru in Plate Function.

Software Design

Inainte de orice implementare software, a trebuit sa calibrez celula de incarcare:

1) Am pregatit un obiect cu o greutate cunoscută. Mi-am folosit cântarul de bucătărie și am cântărit o sticluta de parfum (160g).

2) Am incarcat următorul cod pe placa Arduino. Am scris următorul cod ținând cont de instrucțiunile de calibrare a celulei de sarcină furnizate de documentația bibliotecii.

3) Am determinat factorul de calibrare si l-am inclus in codul proiectului meu.

```
#include <HX711_ADC.h>
#if defined(ESP8266)|| defined(ESP32) || defined(AVR)
#include <EEPROM.h>
#endif

//pins:
const int HX711_dout = 2; //mcu > HX711 dout pin
const int HX711_sck = 3; //mcu > HX711 sck pin

//HX711 constructor:
HX711_ADC LoadCell(HX711_dout, HX711_sck);

const int calVal_eepromAdress = 0;
unsigned long t = 0;

void setup() {
  Serial.begin(57600); delay(10);
  Serial.println();
  Serial.println("Starting...");

  LoadCell.begin();
  //LoadCell.setReverseOutput(); //uncomment to turn a negative output value
to positive
  unsigned long stabilizingtime = 2000; // preciscion right after power-up
can be improved by adding a few seconds of stabilizing time
  boolean _tare = true; //set this to false if you don't want tare to be
performed in the next step
  LoadCell.start(stabilizingtime, _tare);
```

```
  if (LoadCell.getTareTimeoutFlag() || LoadCell.getSignalTimeoutFlag()) {
    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");
    while (1);
  }
  else {
    LoadCell.setCalFactor(1.0); // user set calibration value (float),
initial value 1.0 may be used for this sketch
    Serial.println("Startup is complete");
  }
  while (!LoadCell.update());
  calibrate(); //start calibration procedure
}

void loop() {
  static boolean newDataReady = 0;
  const int serialPrintInterval = 0; //increase value to slow down serial
print activity

  // check for new data/start next conversion:
  if (LoadCell.update()) newDataReady = true;

  // get smoothed value from the dataset:
  if (newDataReady) {
    if (millis() > t + serialPrintInterval) {
      float i = LoadCell.getData();
      Serial.print("Load_cell output val: ");
      Serial.println(i);
      newDataReady = 0;
      t = millis();
    }
  }

  // receive command from serial terminal
  if (Serial.available() > 0) {
    char inByte = Serial.read();
    if (inByte == 't') LoadCell.tareNoDelay(); //tare
    else if (inByte == 'r') calibrate(); //calibrate
    else if (inByte == 'c') changeSavedCalFactor(); //edit calibration value
manually
  }

  // check if last tare operation is complete
  if (LoadCell.getTareStatus() == true) {
    Serial.println("Tare complete");
  }

}

void calibrate() {
  Serial.println("***");
  Serial.println("Start calibration:");
```

```
  Serial.println("Place the load cell an a level stable surface.");
  Serial.println("Remove any load applied to the load cell.");
  Serial.println("Send 't' from serial monitor to set the tare offset.");

  boolean _resume = false;
  while (_resume == false) {
    LoadCell.update();
    if (Serial.available() > 0) {
      if (Serial.available() > 0) {
        char inByte = Serial.read();
        if (inByte == 't') LoadCell.tareNoDelay();
      }
    }
    if (LoadCell.getTareStatus() == true) {
      Serial.println("Tare complete");
      _resume = true;
    }
  }

  Serial.println("Now, place your known mass on the loadcell.");
  Serial.println("Then send the weight of this mass (i.e. 100.0) from serial
monitor.");

  float known_mass = 0;
  _resume = false;
  while (_resume == false) {
    LoadCell.update();
    if (Serial.available() > 0) {
      known_mass = Serial.parseFloat();
      if (known_mass != 0) {
        Serial.print("Known mass is: ");
        Serial.println(known_mass);
        _resume = true;
      }
    }
  }

  LoadCell.refreshDataSet(); //refresh the dataset to be sure that the known
mass is measured correct
  float newCalibrationValue = LoadCell.getNewCalibration(known_mass); //get
the new calibration value

  Serial.print("New calibration value has been set to: ");
  Serial.print(newCalibrationValue);
  Serial.println(", use this as calibration value (calFactor) in your
project sketch.");
  Serial.print("Save this value to EEPROM adress ");
  Serial.print(calVal_eepromAdress);
  Serial.println("? y/n");

  _resume = false;
```

```
  while (_resume == false) {
    if (Serial.available() > 0) {
      char inByte = Serial.read();
      if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.begin(512);
#endif
        EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.commit();
#endif
        EEPROM.get(calVal_eepromAdress, newCalibrationValue);
        Serial.print("Value ");
        Serial.print(newCalibrationValue);
        Serial.print(" saved to EEPROM address: ");
        Serial.println(calVal_eepromAdress);
        _resume = true;

      }
      else if (inByte == 'n') {
        Serial.println("Value not saved to EEPROM");
        _resume = true;
      }
    }
  }

  Serial.println("End calibration");
  Serial.println("***");
  Serial.println("To re-calibrate, send 'r' from serial monitor.");
  Serial.println("For manual edit of the calibration value, send 'c' from
serial monitor.");
  Serial.println("***");
}

void changeSavedCalFactor() {
  float oldCalibrationValue = LoadCell.getCalFactor();
  boolean _resume = false;
  Serial.println("***");
  Serial.print("Current value is: ");
  Serial.println(oldCalibrationValue);
  Serial.println("Now, send the new value from serial monitor, i.e. 696.0");
  float newCalibrationValue;
  while (_resume == false) {
    if (Serial.available() > 0) {
      newCalibrationValue = Serial.parseFloat();
      if (newCalibrationValue != 0) {
        Serial.print("New calibration value is: ");
        Serial.println(newCalibrationValue);
        LoadCell.setCalFactor(newCalibrationValue);
        _resume = true;
      }
```

```
    }
  }
  _resume = false;
  Serial.print("Save this value to EEPROM adress ");
  Serial.print(calVal_eepromAdress);
  Serial.println("? y/n");
  while (_resume == false) {
    if (Serial.available() > 0) {
      char inByte = Serial.read();
      if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.begin(512);
#endif
        EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.commit();
#endif
        EEPROM.get(calVal_eepromAdress, newCalibrationValue);
        Serial.print("Value ");
        Serial.print(newCalibrationValue);
        Serial.print(" saved to EEPROM address: ");
        Serial.println(calVal_eepromAdress);
        _resume = true;
      }
      else if (inByte == 'n') {
        Serial.println("Value not saved to EEPROM");
        _resume = true;
      }
    }
  }
  Serial.println("End change calibration value");
  Serial.println("***");
}
```

Descriere detaliată a codului:

Biblioteci incluse:

* LiquidCrystal.h: pentru controlul afișajului LCD.

* HX711_ADC.h: pentru citirea datelor de la senzorul de greutate HX711.

```
#include <LiquidCrystal.h>
#include <HX711_ADC.h>
```

Definirea pinilor și inițializarea componentelor:

* Pinii DOUT_PIN și SCK_PIN sunt definiți pentru senzorul HX711.

* Obiectul LoadCell este creat pentru a interacționa cu senzorul HX711.

* Pinii pentru afișajul LCD sunt definiți (rs, en, d4, d5, d6, d7), iar obiectul lcd este creat.

* Pinii buttonPin, ledPin și plateButton sunt definiți pentru butoanele de control și LED.

```
// Define HX711 pins
#define DOUT_PIN 2
#define SCK_PIN 3

// Initialize the HX711
HX711_ADC LoadCell(DOUT_PIN, SCK_PIN);

// Initialize the LCD library with the numbers of the interface pins
const int rs = 13, en = 12, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

const int buttonPin = 8;  // Define the pin for the button
const int ledPin = 9;
const int plateButton= 10;
```

Funcția setup:

* Setează pinul ledPin ca ieșire și pinurile buttonPin și plateButton ca intrări. * Inițializează comunicarea serială la 57600 baud. * Inițializează conexiunea la senzorul HX711 și setează factorul de calibrare. * Resetează senzorul la zero. * Inițializează afișajul LCD și afișează un mesaj de bun venit timp de 3 secunde.

```
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  pinMode(plateButton, INPUT);

  Serial.begin(57600);

  // Start the connection to HX711
  LoadCell.begin();
  LoadCell.start(1000); // Allow the load cells to stabilize for 1000ms
  LoadCell.setCalFactor(-386.62); // Set the calibration factor
  // -386.92
  // -385.88
```

```
  LoadCell.tare(); // Reset the scale to zero

  // Initialize the LCD
  lcd.begin(16, 2);
  lcd.setCursor(1, 0); // Set cursor to the first row
  lcd.print("Cook Pro"); // Print out to LCD
  lcd.setCursor(4, 1); // Set cursor to the second row
  lcd.print("Measure"); // Print out to LCD
  delay(3000);
  lcd.clear();
}
```

Funcția loop:

* Citirea și afișarea greutății măsurate de senzorul HX711. * Verifică dacă greutatea este stabilă și dacă este prima rulare. * Afișează greutatea pe afișajul LCD, fie ca greutate simplă, fie ca greutate ajustată pentru un suport (tavă). * Gestionează resetarea și funcția de suport prin intermediul butoanelor: * Dacă buttonPin este apăsat, resetează greutatea și afișează mesajul de resetare. * Dacă plateButton este apăsat, activează funcția de suport și afișează mesajul corespunzător.

```
void loop() {
  static float lastWeight = 0;
  static bool isStable = false;
  static bool isFirstRun = true;
  static bool plate = false;
  static float plateWeight = 0;
  static int ledBrightness = 0; // Variable to store LED brightness

  LoadCell.update();
  float weight = LoadCell.getData();

  Serial.print("Weight: ");
  Serial.print(weight, 2); // Print the weight with 2 decimal places
  Serial.println(" g");

  // Retrieve data from the load cell
  if (isFirstRun) {
    lastWeight = 0.0; // Set lastWeight to 0.0 for the first run
    isFirstRun = false; // Set isFirstRun to false after the first run
  }

  // Check if weight is stable
  if (abs(weight - lastWeight) < 0.05) { // Adjust the threshold as needed
    isStable = true;
  } else {
    isStable = false;
  }
```

```
  if (weight < 0) {
    weight = 0;
  } else if (weight > 0 && weight < 1) {
    weight = 0;
  }

  if (isStable) {
    if (!plate) {
      lcd.clear();
      lcd.setCursor(0, 0); // Set cursor to the first row
      lcd.print("Weight: ");
      lcd.print(weight); // Print the weight with 2 decimal places
      lcd.print(" g");
    } else {
      if (weight < 20 && weight > 15) {
        plateWeight = weight;
        weight = 0;
      }
      if (weight > plateWeight) {
        weight = weight - plateWeight;
      }
      lcd.clear();
      lcd.setCursor(0, 0); // Set cursor to the first row
      lcd.print("Weight: ");
      lcd.print(weight); // Print the weight with 2 decimal places
      lcd.print(" g");

      lcd.setCursor(0, 1); // Adjust cursor position
      lcd.print("With plate");
    }
  }

  lastWeight = weight;

  // Check if the button is pressed
  if (digitalRead(buttonPin) == HIGH) {
    analogWrite(ledPin, 0); // Turn off the LED
    Serial.println("Reset button pressed");
    plate = false;
    weight = 0;
    lcd.clear();
    lcd.setCursor(0, 1); // Set cursor to the second row
    lcd.print("    Reset...    ");
    LoadCell.start(1000);
    LoadCell.tare(); // Reset the scale to zero
    delay(1000); // Allow some time for the reset to stabilize
    lcd.clear();
  }

  // Plate button functionality with PWM
  if (digitalRead(plateButton) == HIGH) {
```

```
    Serial.println("PLATE FUNCTION");
    plate = true;
    lcd.clear();
    lcd.setCursor(0, 1); // Set cursor to the second row
    lcd.print(" Plate function ");
    LoadCell.start(1000);
    LoadCell.tare(); // Reset the scale to zero
    delay(1000); // Allow some time for the reset to stabilize
    lcd.clear();

    // Gradually increase LED brightness using PWM
    for (int brightness = 0; brightness <= 255; brightness += 5) {
      analogWrite(ledPin, brightness);
      delay(10);
    }
  }

  // Decrease LED brightness over time
  if (ledBrightness > 0) {
    ledBrightness -= 5;
    if (ledBrightness < 0) {
      ledBrightness = 0;
    }
    analogWrite(ledPin, ledBrightness);
  }

  delay(100); // Update every 100ms
}
```

Funcționalități cheie:

* Stabilizarea greutății: Codul verifică dacă greutatea citită este stabilă (variația este sub un prag specificat).

* Resetare: Butonul de resetare setează greutatea la zero și resetează senzorul.

* Funcția de suport: Un buton suplimentar activează o funcție care ajustează greutatea pentru a scădea greutatea suportului (tăvii).

* Acest program este util pentru aplicații unde este necesară măsurarea precisă a greutății, cum ar fi în bucătării pentru măsurarea ingredientelor sau în alte aplicații de cântărire.

Rezultate Obţinute

Am atasat un videoclip in care sunt prezentate functionalitatile proiectului meu:

Concluzii

Am considerat acest proiect extrem de interesant și util în viața de zi cu zi, oferindu-mi satisfacția de a-l realiza de la zero. Deși am întâmpinat câteva dificultăți, în special în partea de calibrare, am reușit să le depășesc. Modulul HX711 este foarte sensibil și necesită o poziționare precisă pentru a măsura corect greutatea. Am rezolvat această problemă adăugând un suport stabil pentru cântar, asigurându-mă că forța aplicată acționează într-o direcție specifică asupra load cell-ului și că toate componentele sunt bine fixate. Această ajustare a fost esențială pentru obținerea unor măsurători precise și consistente.

Download

cantar_bucatarie.ino.zip

Jurnal

- 05.05.2024: Creare pagina wiki
- 13.05.2024: Comandare piese hardware
- 15.05.2024: Crearea schemelor pentru circuit
- 17.05.2024: Asamblarea circuitului hardware
- 19.05.2024: Rezolvare problema LCD ului (Black boxes)
- 20.05.2024: Scriere cod si calibrarea load cell-ului
- 21.05.2024: Adaugare buton de reset
- 22.05.2024: Adaugarea butonului de Plate Function si a LED-ului
- 24.05.2024: Retusarea codului si a circuitului

Bibliografie/Resurse

Resurse hardware

https://forum.arduino.cc/t/10kg-load-cell-tal220-with-hx-711-gives-reading-of-0-all-the-time/461870

https://www.diyengineers.com/2022/05/19/load-cell-with-hx711-how-to-use-with-examples/
https://nicuflorica.blogspot.com/2018/02/cantar-electronic-pentru-greutati-mic.html

Resurse Software

https://randomnerdtutorials.com/arduino-load-cell-hx711

https://www.instructables.com/Tutorial-How-to-Calibrate-and-Interface-Load-Cell-/

Export to PDF