UNIVERSITATEA POLITEHNICA TIMIȘOARA
Facultatea de Automatică și Calculatoare

Sisteme Incorporate

-Sistem sortare mingi dupa culoare-

Kafka Patrik-Franz, 2022-2023

Cuprins:

1.Tema proiectului:

Proiectul presupune o constructie cu o rampa, pe care se vor rostogoli mingi de diferite culori si care vor fi sortate la baza rampei bazate pe culoare iar cele care nu corespund vor fi aruncate de pe rampa. Un motor stepper va fi amplasat in varful rampei pentru a elibera mingile, unul pe la mijloc care va arunca mingile nepotrivite iar ultimul la baza pentru a ghida mingile in locurile corespunzatoare.

Un senzor de culoare va fi amplasat pe rampa pentru a determina culoarea. Stepperele vor fi controlate folosind un shield CNC pentru arduino si driverele corespunzatoare.

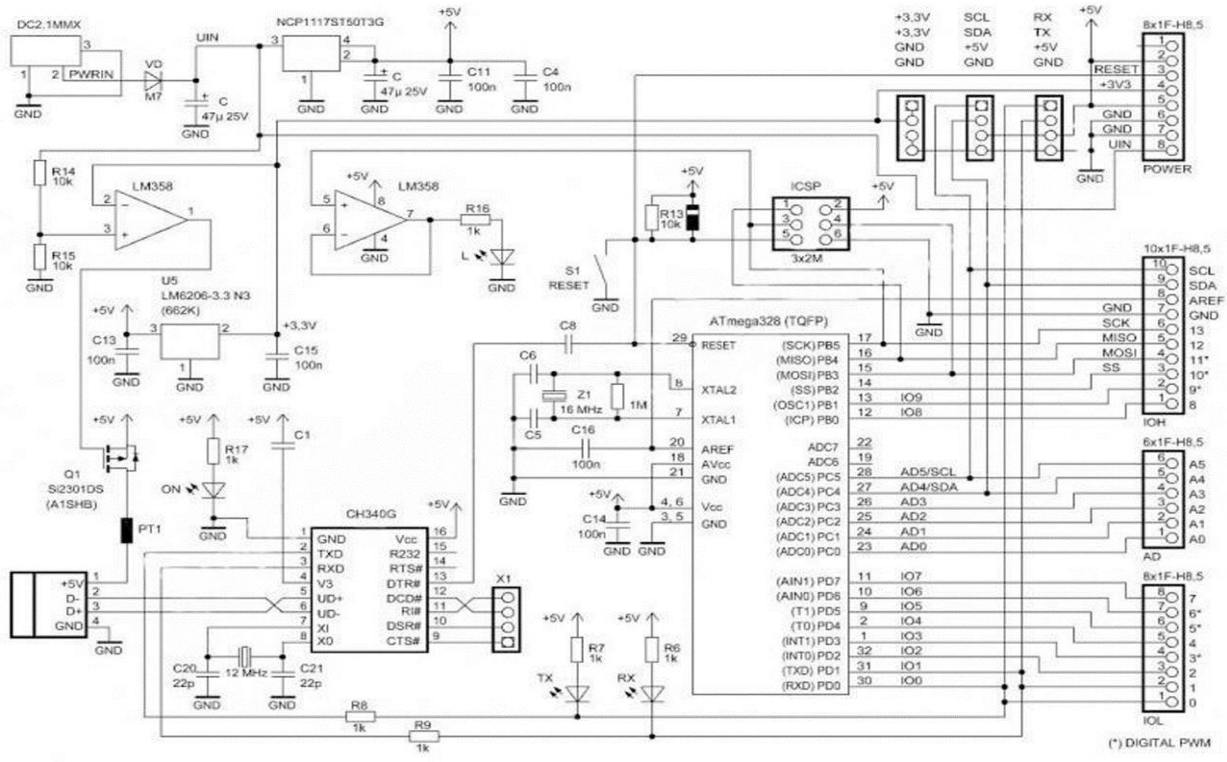
Cu ajutorul datelor prelucrate de la senzorul de culoare se vor actiona celelalte servomotoare pentru actiunile prezentate mai in sus.

Stepper-ul de la baza va directiona bilele in compartimentele potrivite culorii, care pot fi configurate in cod.

Extra, voi adauga un display LCD pentru afisarea datelor

2.Descrierea placilor de dezvolare:

1. Arduino Uno R3:



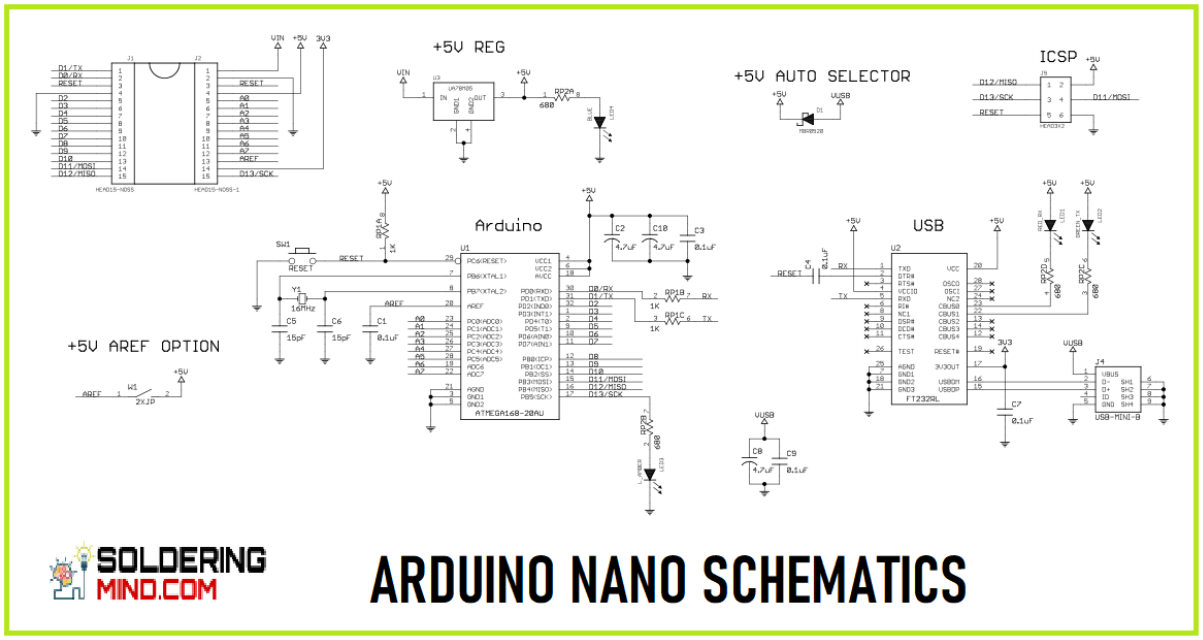
Arduino Uno R3 este o placă de dezvoltare bazată pe microcontrollerul ATmega328. Are 14 pini de intrare/ieșire (dintre care 6 pot fi folosiți ca ieșiri PWM), 6 intrări analog, un oscilator de 16MHz, o conexiune USB, mufă de alimentare, și un buton de reset.

Poate fi alimentat direct de la calculator, de la portul USB, prin intermediul unei baterii de 9V sau a unui alimentator de 9V.

Caracteristici tehnice:

- Microcontroller ATmega328
- Tensiune de operare: 5V
- Tensiune de alimentare recomandată: 7-12V
- Limită de tensiune: 6-20V
- Pini intrare/ieșire digitali: 14 (dintre care 6 pot oferi ieșire PWM)
- Pini analogici de intrare: 6
- Memorie Flash 32 KB
- SRAM 2 KB
- EEPROM 1 KB
- Frecvență de lucru: 16 MHz

2. Arduino nano:



Placa de dezvoltare este echipată cu același micro-controller performant (ATmega328p) de pe Arduino Uno si convertorul USB serial CH340. Avantajul acesteia îl reprezintă dimensiunile reduse, astfel se poate integra în diverse proiecte unde spațiul componentelor este foarte important. Programarea dispozitivului se realizează prin intermediul unui cablu cu mufa mini USB, placa de dezvoltare venind și cu un bootloader.

Caracteristici tehnice:

3. Tensiune de alimentare suportată de limitator: 7 V - 12 V;
4. Tensiune de alimentare: 5 V;
5. Pini Input/Output: 14;
6. Pini ADC: 8 (din cei 14 de Input/Output);
7. Pini PWM: 6 (din cei 14 de Input/Output);
8. Memorie flash: 32 kB / 16 kB (din care 2 kB sunt folosiți de bootloader);
9. Comunicație TWI, SPI și UART;
10. Curent pentru pini Input/Output: 40 mA/pin;
11. Frecvență de funcționare: 16 MHz;
12. Dimensiuni: 45 x 18 mm

- Pentru placa de dezvoltare arduino uno r3 avem microcontrollerul Atmega328P:

ATmega328P este un microcontroler AVR pe 8 biți de înaltă performanță, dar cu consum redus de energie, care este capabil să realizeze cel mai simplu ciclu de execuție a 131 de instrucțiuni puternice datorită arhitecturii sale

avansate RISC. Poate fi găsit în mod obișnuit ca procesor în plăcile Arduino, cum ar fi Arduino Fio și Arduino Uno.

In plus, se bazează pe un microcontroler RISC, combinând 32 KB ISP flash o memorie cu capacitatea de a citi-în-timp-ce-scrie, 1 KB de memorie EEPROM, 2 KB de SRAM, 23 linii E/S de uz general, 32 Înregistrari procese generale, trei cronometre flexibile/contoare în comparație cu, întreruperi internă și externă, programator de tip USART, orientate interfață serială byte de 2 cabluri, SPI port serial, 6-canale 10-bit Converter A/D (8-chanale în TQFP și QFN/MLF packages), "watchdog timer" programabil cu oscilator intern, și cinci moduri de software-ul intern de economisire a energiei selectabil. Dispozitivul funcționează 1,8-5,5 volți.

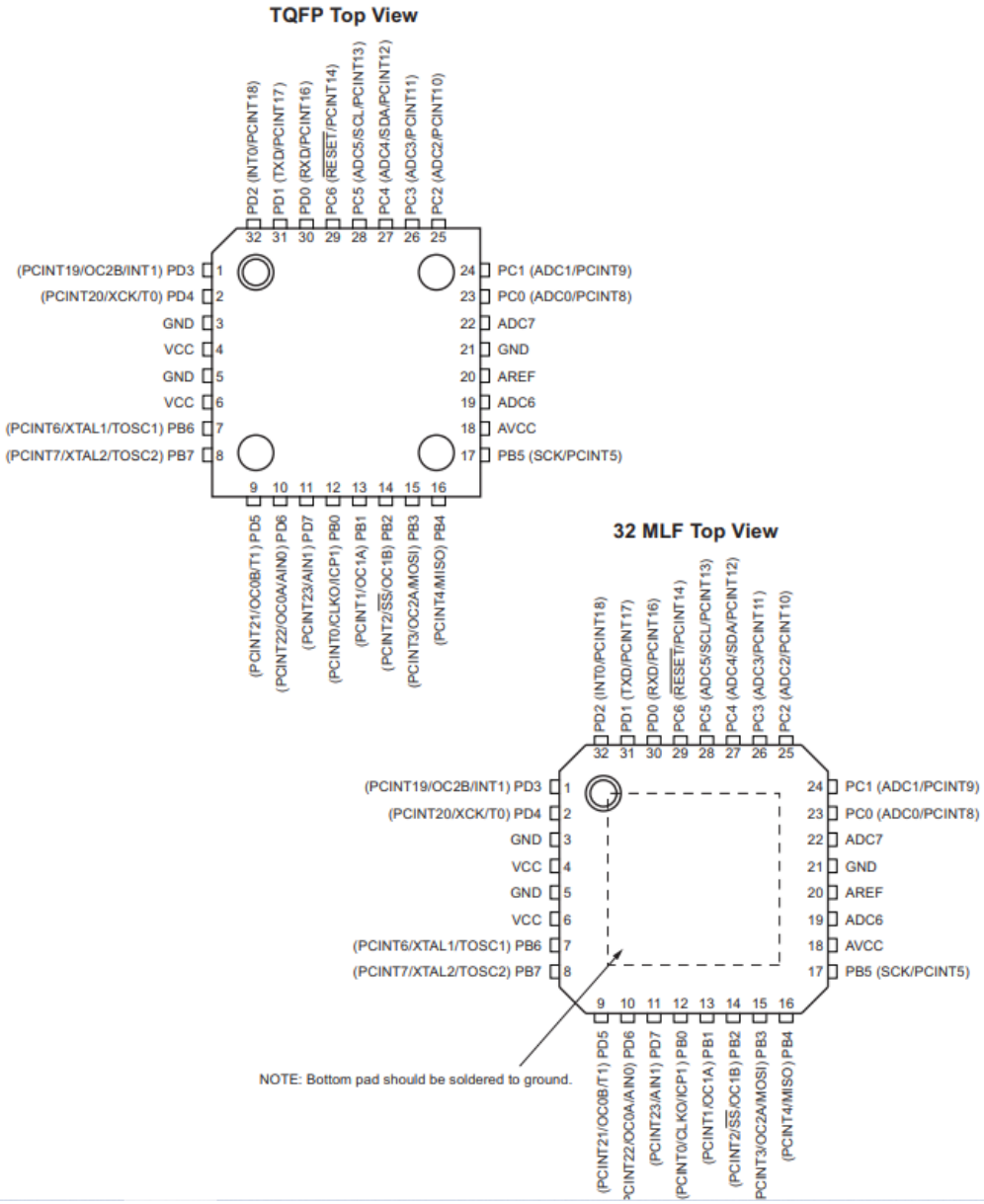
Prin executarea instrucțiuni puternice într-un singur ciclu de ceas, aparatul realizează un răspuns de 1 MIPS

Caracteristici și parametri

Caracteristici:
Segmente de memorie nevolatilă de înaltă rezistență
În memoria programului flash autoprogramabilă a sistemului
Blocare de programare pentru securitatea software-ului
Caracteristici periferice
Două temporizatoare/contor pe 8 biți cu prescaler separat, modul de comparare.
Un temporizator/contor pe 16 biți cu prescaler separat, mod de comparare și mod de captură
Măsurarea temperaturii
USART serial programabil și temporizator watchdog cu oscilator separat pe cip
Caracteristici unice în comparație cu alte microcontrolere (ARM, 8051, PIC):
Resetare la pornire și detectare programabilă a stingerii
Oscilator calibrat intern
Surse de întrerupere externe și interne
Șase moduri de repaus: Idle, reducerea zgomotului ADC, economisire energie, oprire, standby și standby extins

1. Pin Configurations

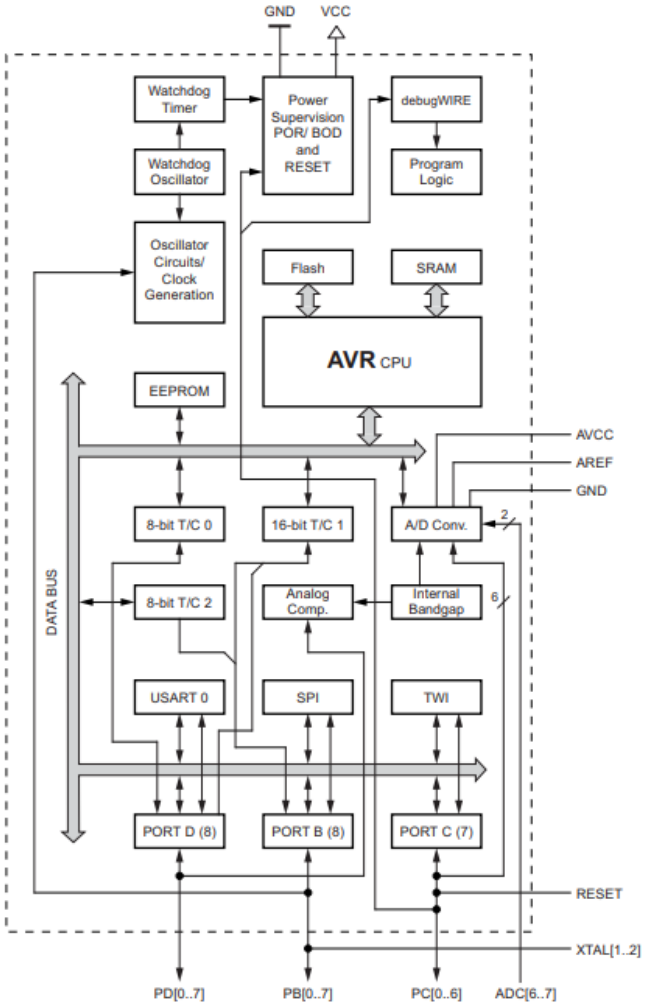
Figure 1-1. Pinout



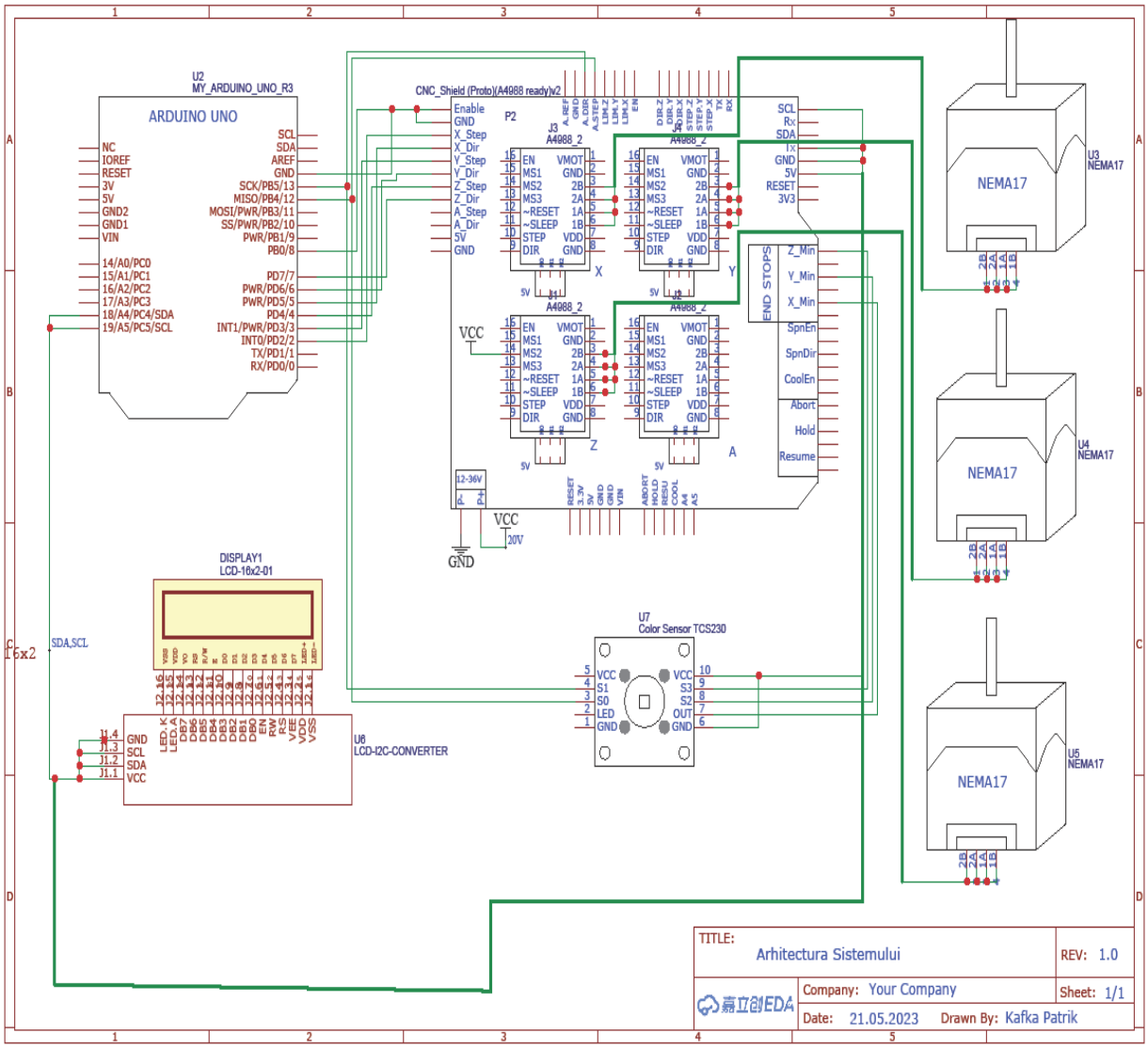
allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



3.Arhitectura sistemului:



- S-a folosit un modul cnc shield v3 pentru arduino uno r3, 3 drivere a 4988, 3 steppere nema17, 1 senzor culoare tcs230, 1 display LCD 16x2 cu interfata I2C.

4.Descrierea detaliata a modulelor microcontrolerului care au fost folosite:

- Modulul I2C, folosit pentru comunicare seriala intre arduino si LCD. Explicatii preluate din articolul : https://profs.info.uaic.ro/~arduino/index.php/Comunicare_I2C

Protocolul Inter Integrated Circuit (I2C) este un protocol creat pentru a permite mai multor circuite integrate “slave” sa comunice cu unul sau mai mute

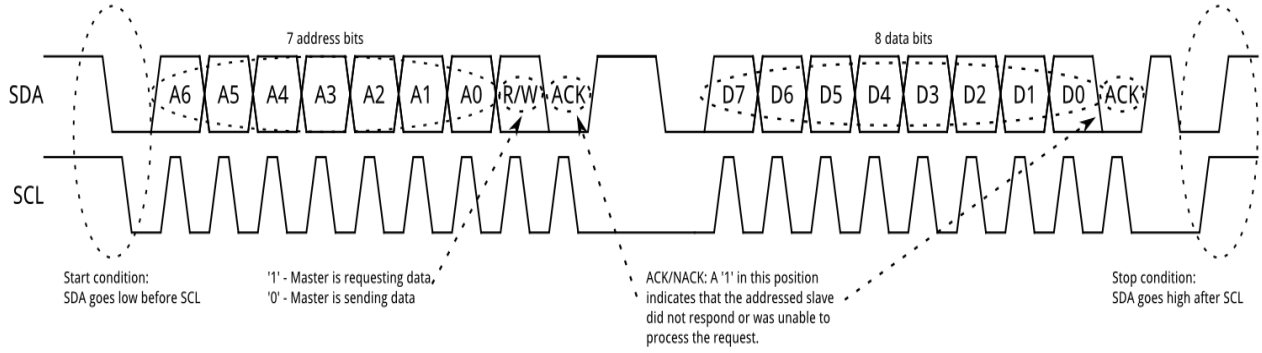
cipuri “master”. Acest tip de comunicare poate fi folosit doar pe distante mici de comunicare si asemenea protocolului UART are nevoie doar de 2 fire de semnal pentru a trimite/primii informatii.

Fiecare bus *I2C* este compus din doua semnale: *SCL* (semnalul de ceas) si *SDA* (semnalul de date). Semnalul de ceas este intotdeauna generat de bus-ul masterul curent. Fiecare bus *I2C* poate suporta pana la 112 dispozitive, iar toate dispozitivele trebuie sa distribuie *GND*.

Spre deosebire de alte metode de comunicare, precum *UART*, magistrala *I2C* este de tip “open drain”, ceea ce inseamna ca poate trage o anumita linie de semnal in 0 logic, dar nu o pot conduce spre 1 logic. Asadar, se elimina problema de "bus contention”, unde un dispozitiv incearca sa traga una dintre linii in starea "high” in timp ce altul o aduce in "low”, eliminand posibilitatea de a distruge componente. Fiecare linie de semnal are un rezistor pull-up pe ea, pentru a putea readuce semnalul pe "high” cand nici un alt dispozitiv nu cere „low”.

Mesajele transmise sunt impartite in doua tipuri de cadre (frames): cadre de date (contin mesaje pe 8 biti directionate de la dispozitivul de tip "master" la cel de tip "slave" si invers) si cadre de adresa (contine adresa dispozitivul de tip "slave", la care dispozitivul de tip "master" trimite mesajul).

Datele sunt puse pe linia SDA dupa ce SCL ajunge la nivel "low" si sunt esantionate cand SCL ajunge "high". Timpul intre nivelul de ceas si operatiile de citire/scriere este definit de dispozitivele conectate pe magistrala si va varia de la cip la cip.



Conditia de start

Pentru a initia cadrul de adresa, dispozitivul "master" lasa SCL "high" si pune SDA pe "low", astfel toate dispozitivele "slave" sunt pregatite pentru a receptiona date.

Daca doua sau mai multe dispozitive "master" doresc sa-si asume bus-ul la un moment dat, primul dispozitiv care ajunge de la "high" la "low" primul, va prelua controlul bus-ului.

Cadrul de adresa

Cadrul de adresa este mereu primul atunci cand o noua comunicare incepe. Mai intai se vor trimite sincron bitii adresei (primul fiind cel mai semnificativ), apoi se va transmite un semnal de tip R/W pe biti. Bitul 9 al cadrului este reprezentat de bitul NACK/ACK. Acesta este cazul pentru ambele cadre (de adrese si de date).

Dupa ce primii 8 biti ai cadrului sunt transmisi, dispozitivului receptor ii este dat controlul asupra SDA-ului. Daca acest dispozitiv nu schimba in 0 logic linia SDA inainte de al noualea puls de ceas, se poate deduce ca acesta nu a primit datele, ori nu a stiut sa interpreteze mesajul receptionat. In acest caz, schimbul de date se opreste si ramane la latitudinea dispozitivului "master" cum va proceda mai departe.

Cadrul de date

Dupa ce cadrul de adresa a fost tramsis, datele/mesajele pot incepe a fi trasmise. Dispozitivul "master" va continua sa genereze impulsuri de ceas la un interval regulat, iar datele vor fi plasate pe SDA, fie de dispozitivul "master" fie de cel "slave", in functie de bitii R/W transmisi in cadrul anterior. Numarul de cadre de date este arbitar.

Conditia de oprire

De indata ce toate cadrele au fost trimise, dispozitivul "master" va genera o conditie de oprire. Conditiile de oprire sunt definite de tranzitia de la "low" la "high" (0 -> 1) pe SDA, dupa p tranzitie 0 -> 1 pe SCL cu SCL pe "high".

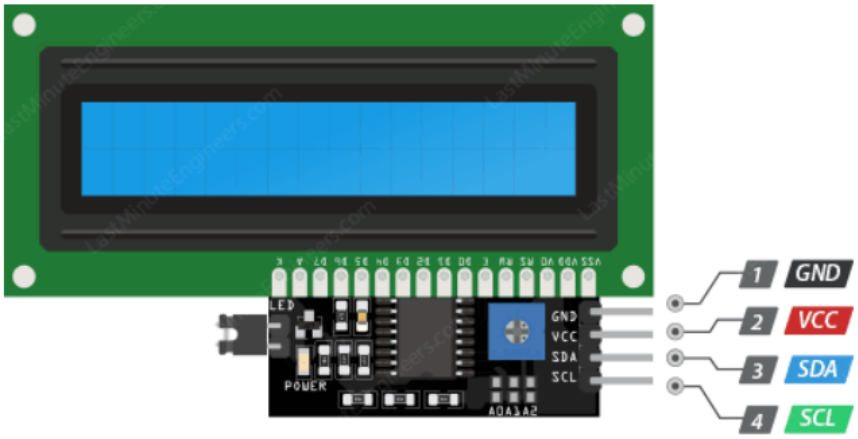
In timpul operatiei de scriere, valoarea din SDA nu ar trebuie sa se schimbe cand SCL este "high" pentru a evita conditiile false de oprire.

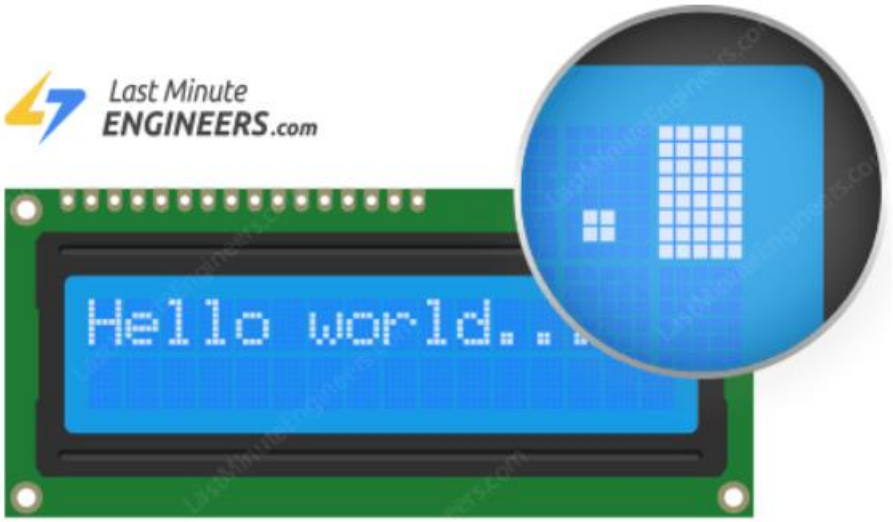
5.Explicarea functionarii sistemului de afisaj

Display-ul LCD (Liquid Crystal Display).
Display-ul folosit este unul 16x2 cu interfata I2C, acesta poate afisa 32 de caractere ascii pe cele 2 linii.

I2C LCD Display Pinout

The I2C LCD Display has only four pins. The following is the pinout:





Se poate vedea ca fiecare caracter este reprezentat prin o matrice 8x5 pe LCD.

Adaptorul I2C converteste datele primite serial de la arduino, in date paralele pentru display.
Daca avem mai multe dispozitive conectate la I2C, trebuie sa ne asiguram ca nu existe conflicte intre adrese.Putem afla adresa noastra prin folosirea unui program examplu din biblioteca Wire, care se gaseste la sectiunea exemple in arduino ide.

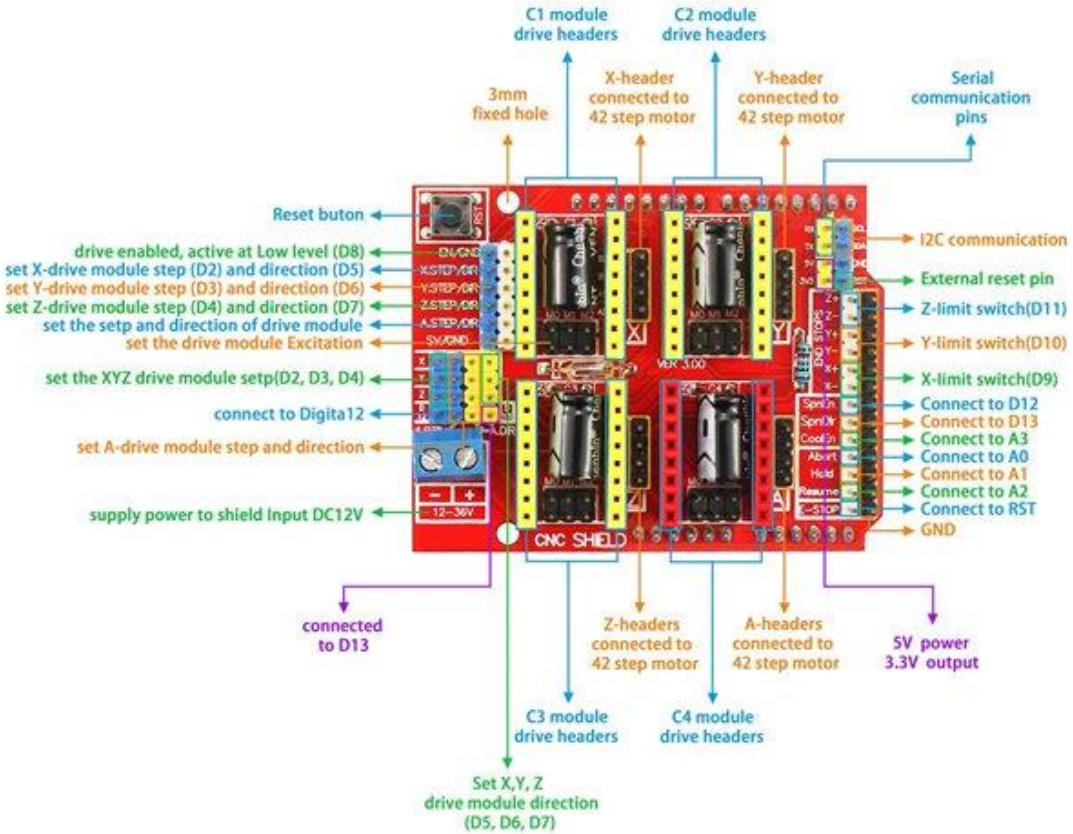
Conectarea unui LCD I2C este mult mai simplă decât conectarea unui LCD standard. Trebuie doar să conectam patru pini.

Începem prin a conecta pinul VCC la ieșirea de 5V a Arduino și pinul GND la masă.

Acum am rămas cu pinii care sunt folosiți pentru comunicarea I2C. Rețineți că fiecare placă Arduino are pini I2C diferiți care trebuie conectați corect. Pe plăcile Arduino cu aspect R3, SDA (linia de date) și SCL (linia de ceas) se află pe anteturile pinului aproape de pinul AREF. Ele sunt denumite și A5 (SCL) și A4 (SDA).

6.Descrierea tehnica a senzorilor si actuatorilor folositi

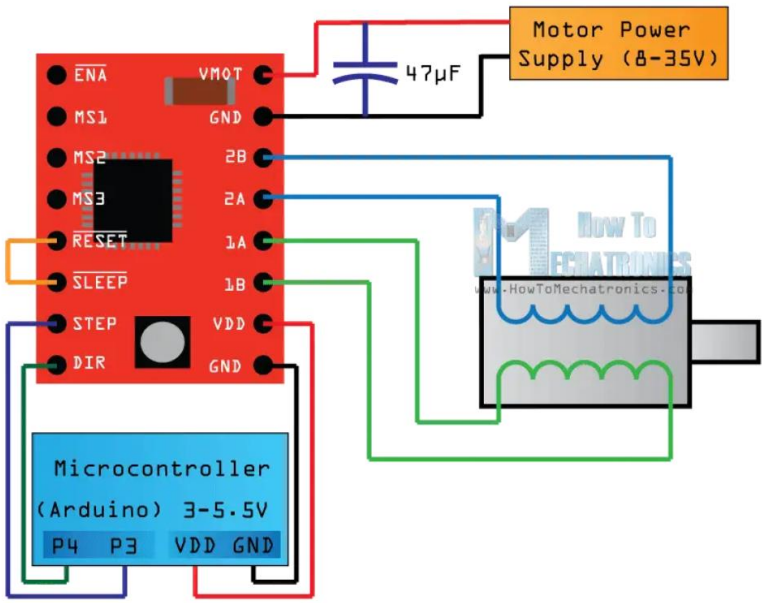
- Cnc shield V3



Placa CNC CNC Shield V3 este un hardware open source folosit pentru controlul motoarelor pas cu pas. Permite controlul simultan al 4 motoare. Folosește module de control stepper A4988 detașabile. Motoarele pas cu pas sunt conectate prin intermediul conectorilor cu 4 pini. Tensiunea de alimentare este de 12-36 V. Motorul pas cu pas este ideal pentru proiecte care necesită rotație controlată. Motoarele pas cu pas recomandate pentru utilizate sunt proiectate conform standardului NEMA 17.

Cu toate acestea, hardware-ul poate fi folosit si in alte scopuri in afara de aplicatii CNC. Cu ajutorul acestui shield putem controla pana la 4 motoare pas cu pas pentru diferite aplicatii.

- Driver-ul pentru motoare pas cu pas A4988



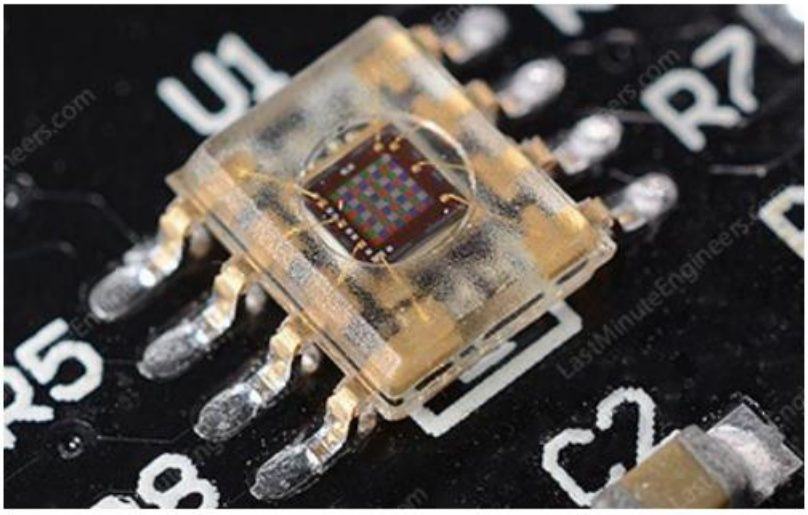
Este un driver capabil de microstepping cu un traductor incorporat cu utilizare usoara. Poate furniza pana la 35V si +-2A, include un regulator de curent. Translatorul functioneaza astfel: prin transmiterea unui impuls la pinul STEP, motorul se va deplasa cu un micropas, in functie de configuratie. In modul pas intreg se va deplasa cu 1.8 grade.

- Motorul pas cu pas nema17:
  Se recomanda un driver pentru controlarea lui.

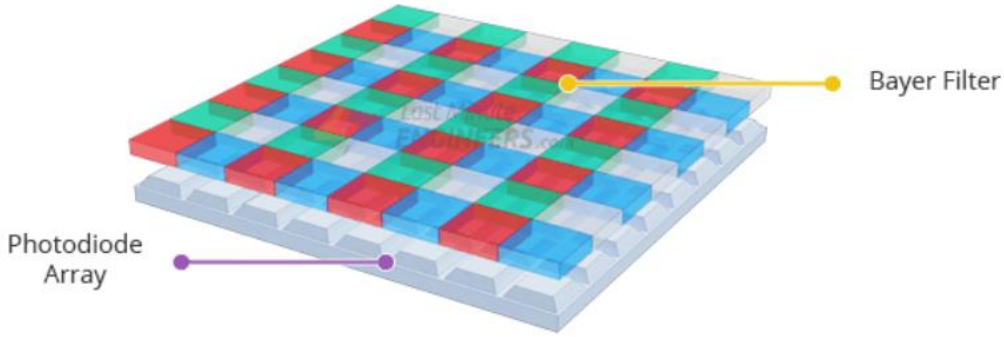
Motorul pas cu pas este un motor de curent continuu fără perii, în care rotația este divizată într-un anumit număr de pași, care rezultă din construcția motorului. În mod standard, o rotație completă a arborelui cu 360 este divizată

în 200 de pași, ceea ce înseamnă că arborele efectuează un nou pas la fiecare 1.8 grade.

- Senzorul de culoare TCS230:



Senzorul functioneaza bazat pe un set de 3 matrici de fotodiode de culoare rosie, verde, albastra.

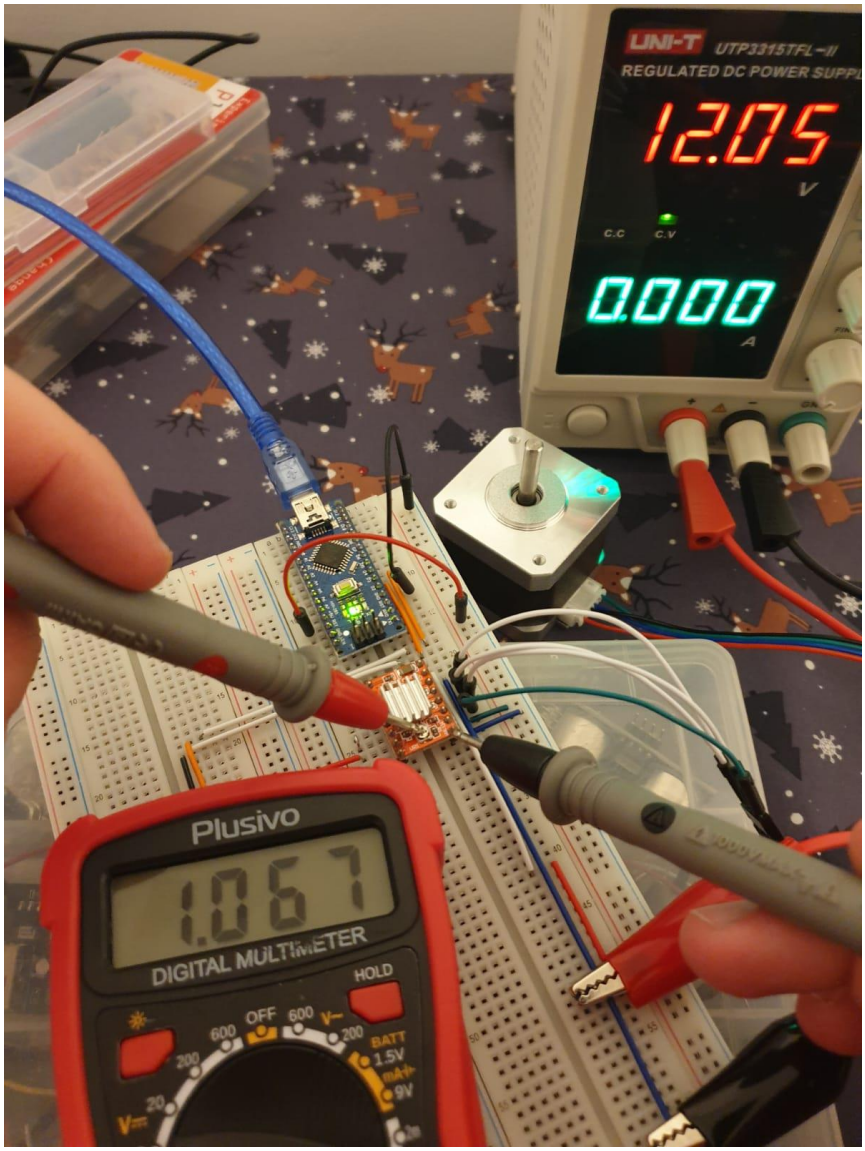


Prin aplicarea semnalelor precum in figura de mai in jos, vom primi la iesire o frecventa convertita intern de senzor pentru fiecare set de fotodiode. In functie de aceste frecvente putem determina culoarea unui obiect.

S2	S3	Photodiode type
LOW	LOW	Red
LOW	HIGH	Blue
HIGH	LOW	Clear (No filter)
HIGH	HIGH	Green

7.Desfasurarea proiectului:

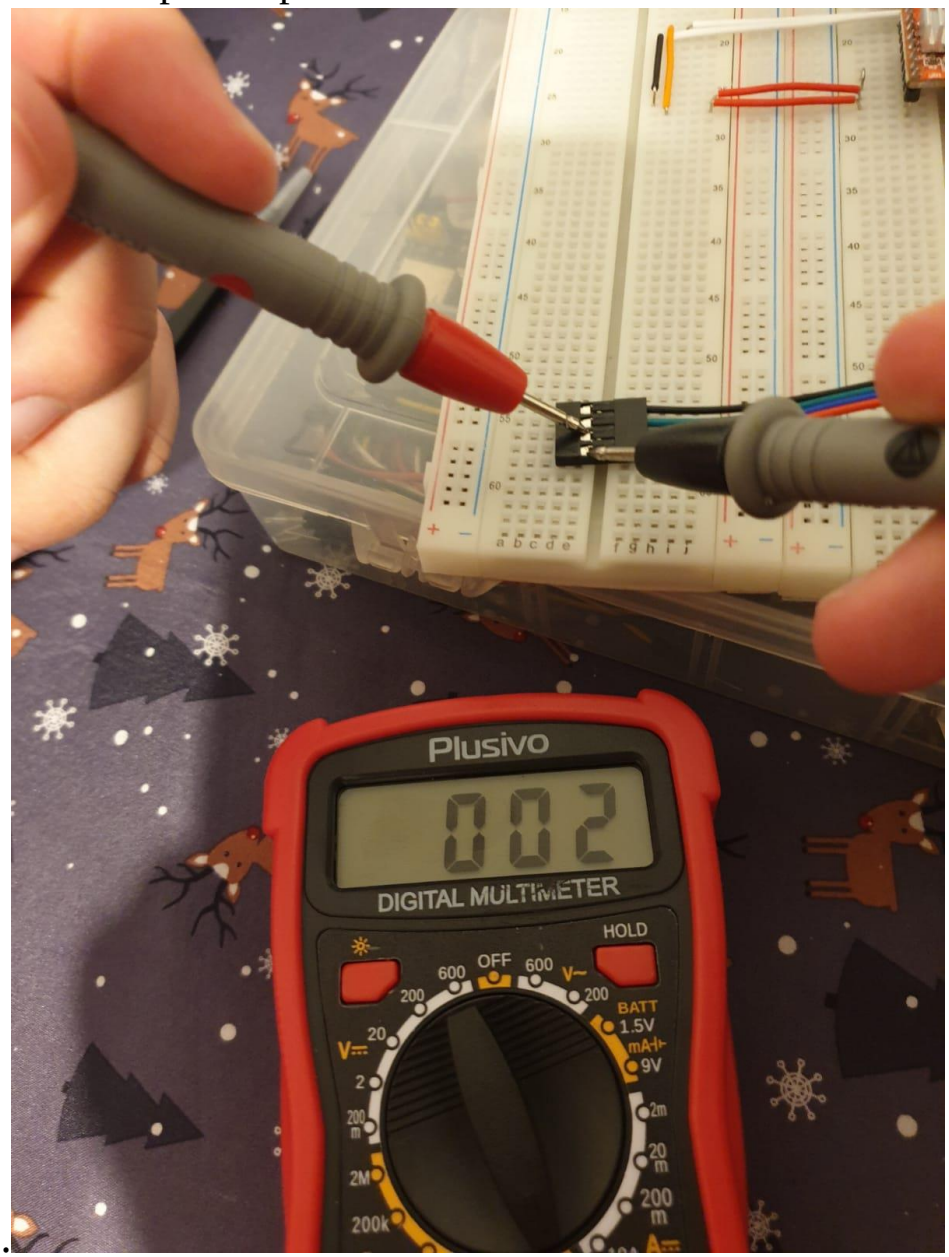
- Intai se va pregati montajul pentru testarea si configurarea driverelor A4988.



Folosind formula data de producator Vref=Imax*8*Rs. Unde Vref este tensiunea masurata cu multimetru intre potentiometru si masa, Imax este curentul maxim pe care il va lasa sa treaca driverul si Rs este un rezistor de 0.1 ohm in cazul acesta.

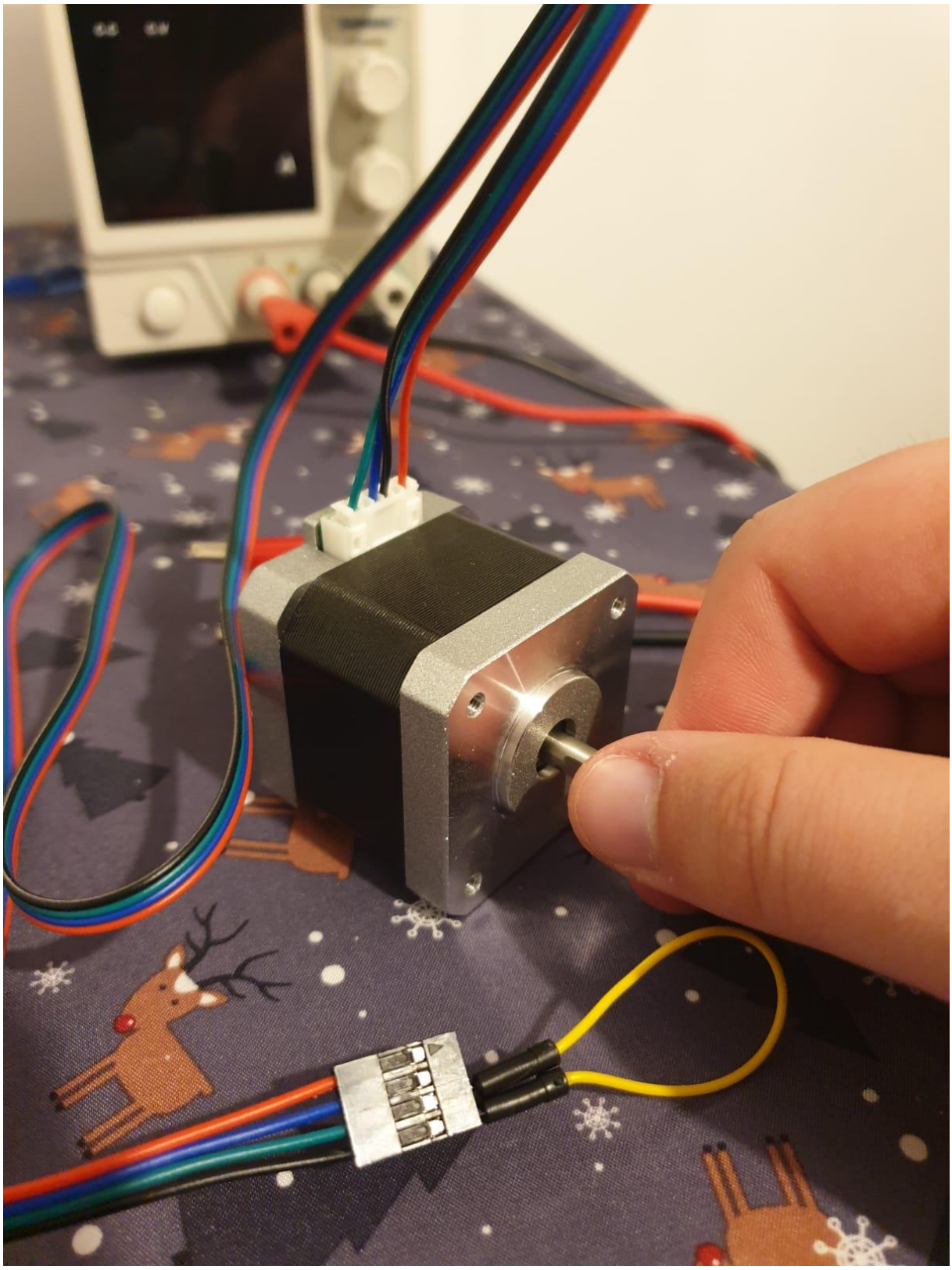
Driverele au fost setate pentru a permite trecerea a cel mult 70% din curentul maxim (1,8A) la care pot functiona mototoarele pas cu pas pentru a evita supraincalzirea circuitelor si a motoarelor.Pentru Vref aproximativ 1v putem atinge aceasta valoare.

- Apoi urmatorul pas a fost idenfiticarea infasurarilor bobinelor din motoarele pas cu pas am folosit 2 metode:

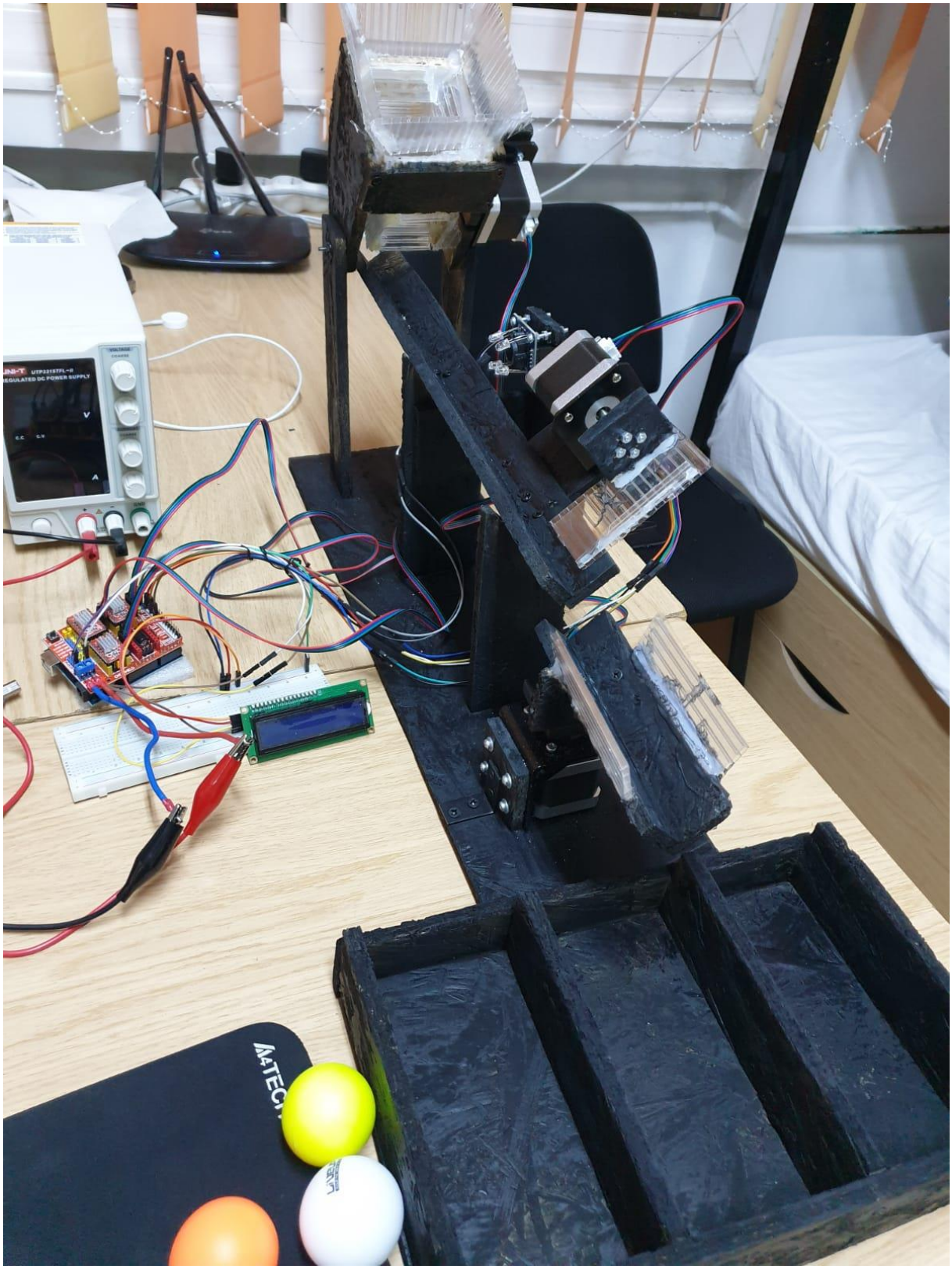
1. 

Cu un multimetru setat pe testarea de continuitate/diodelor, putem identifica infasurarile A,B.

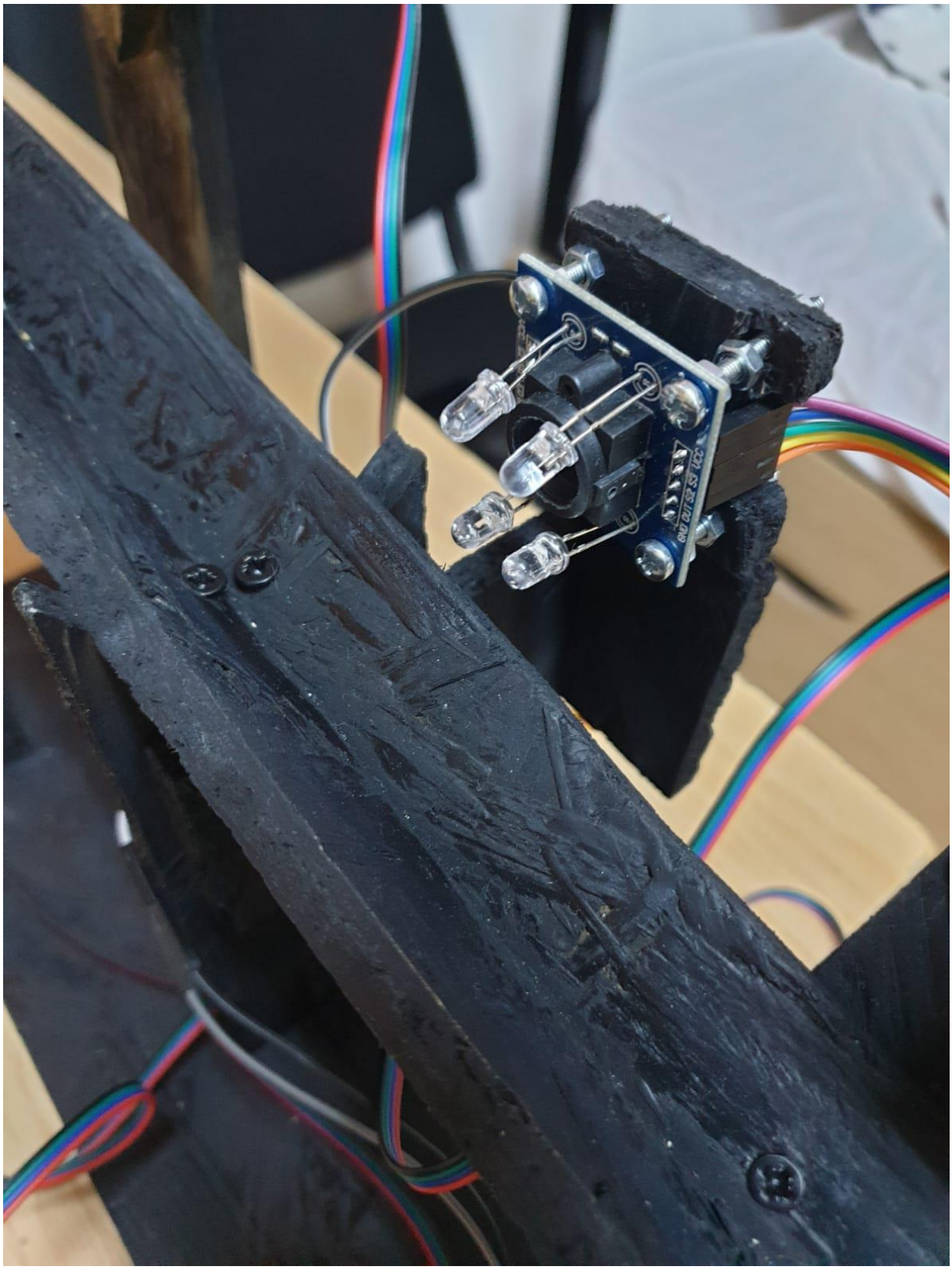
2.O alta metoda este sa legam un fir la terminalele infasurarilor si sa rotim de motor. Cand intampinam o rezistenta mai mare, atunci ne putem da seama ca am scurtcircuitat infasurarea A sau B si asa le putem identifica.



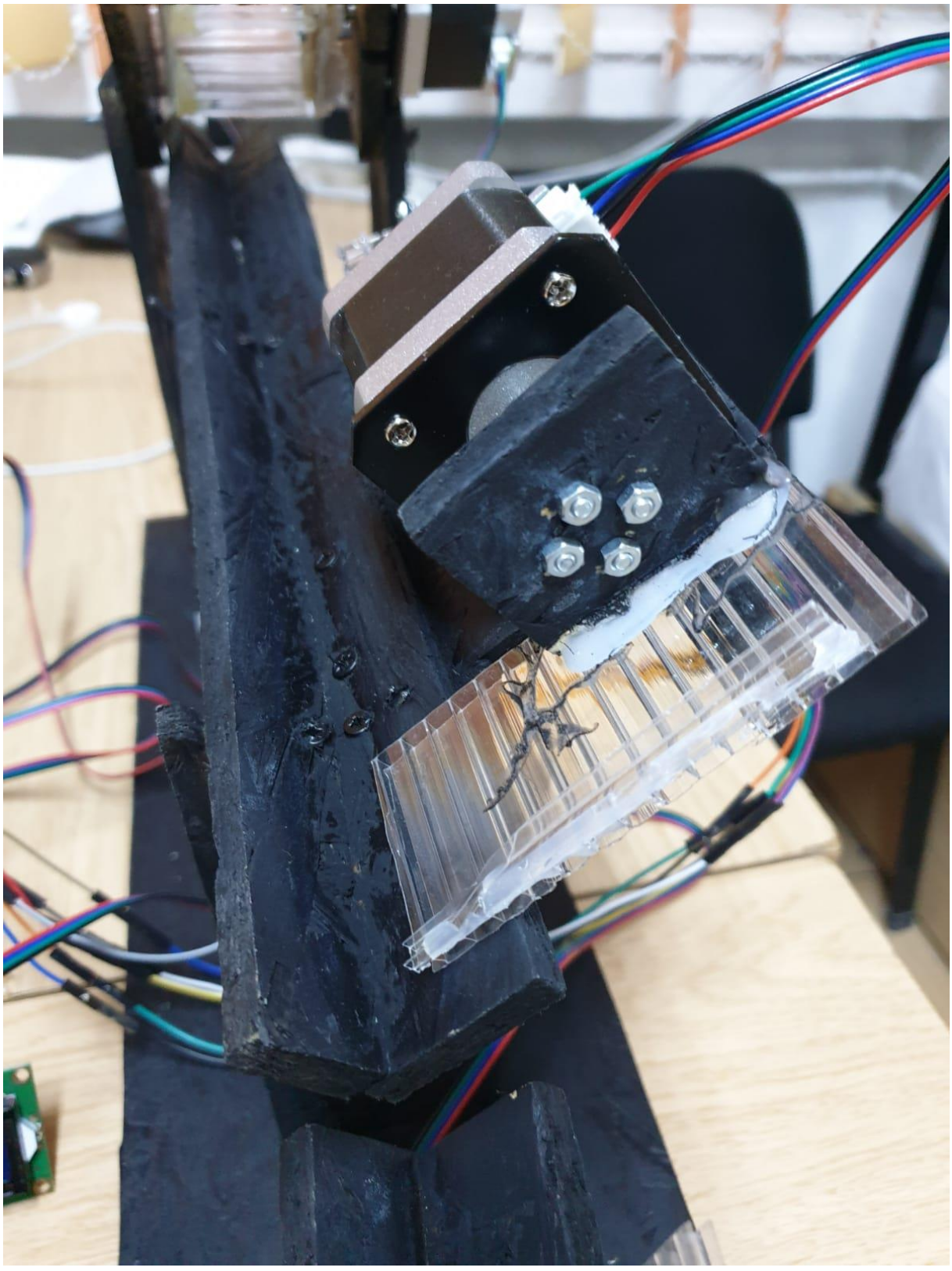
- Urmatorul pas a fost realizarea machetei si montarea senzorilor, actuatorilor:



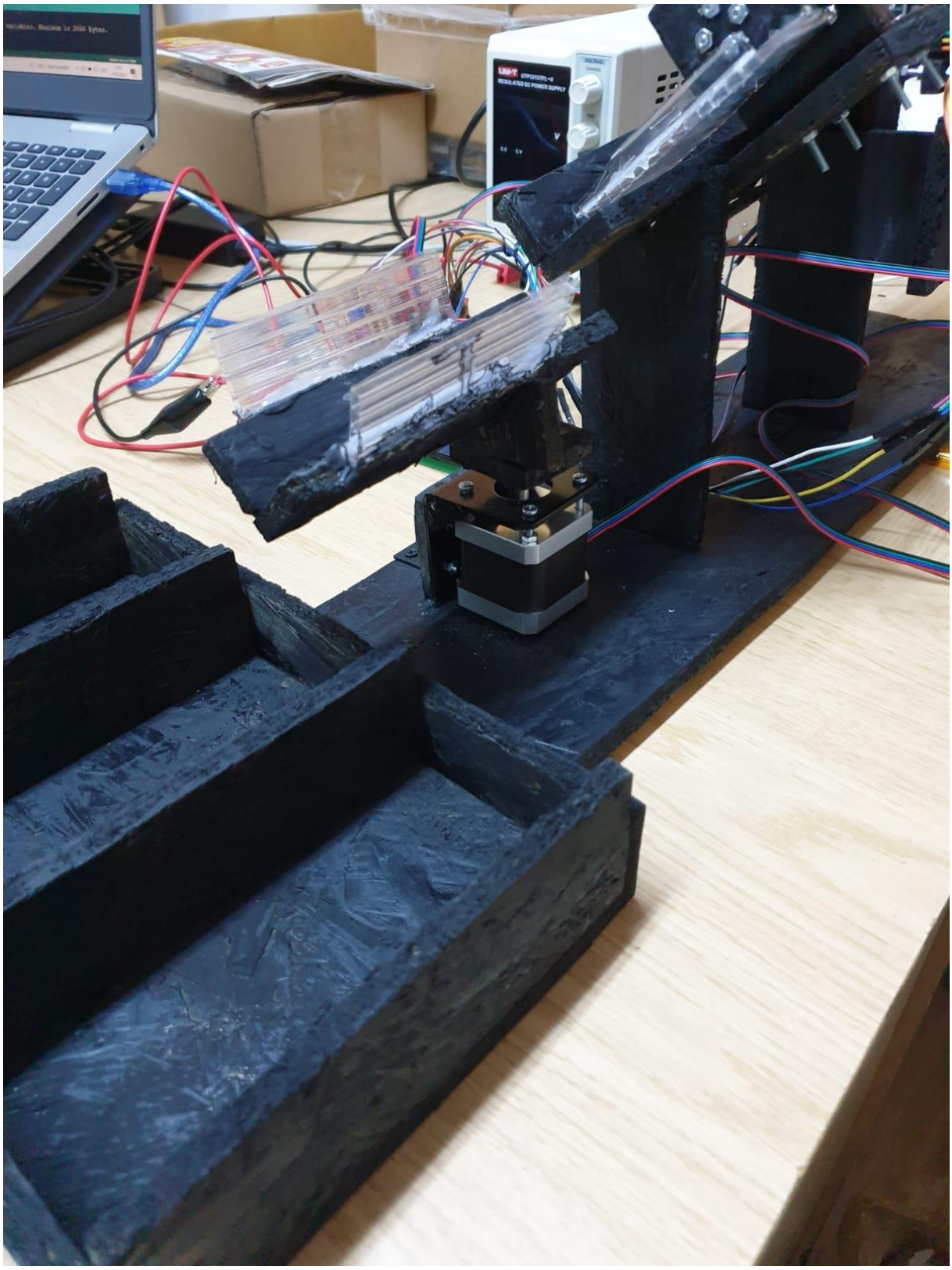
Aici avem fixat senzorul de culoare:



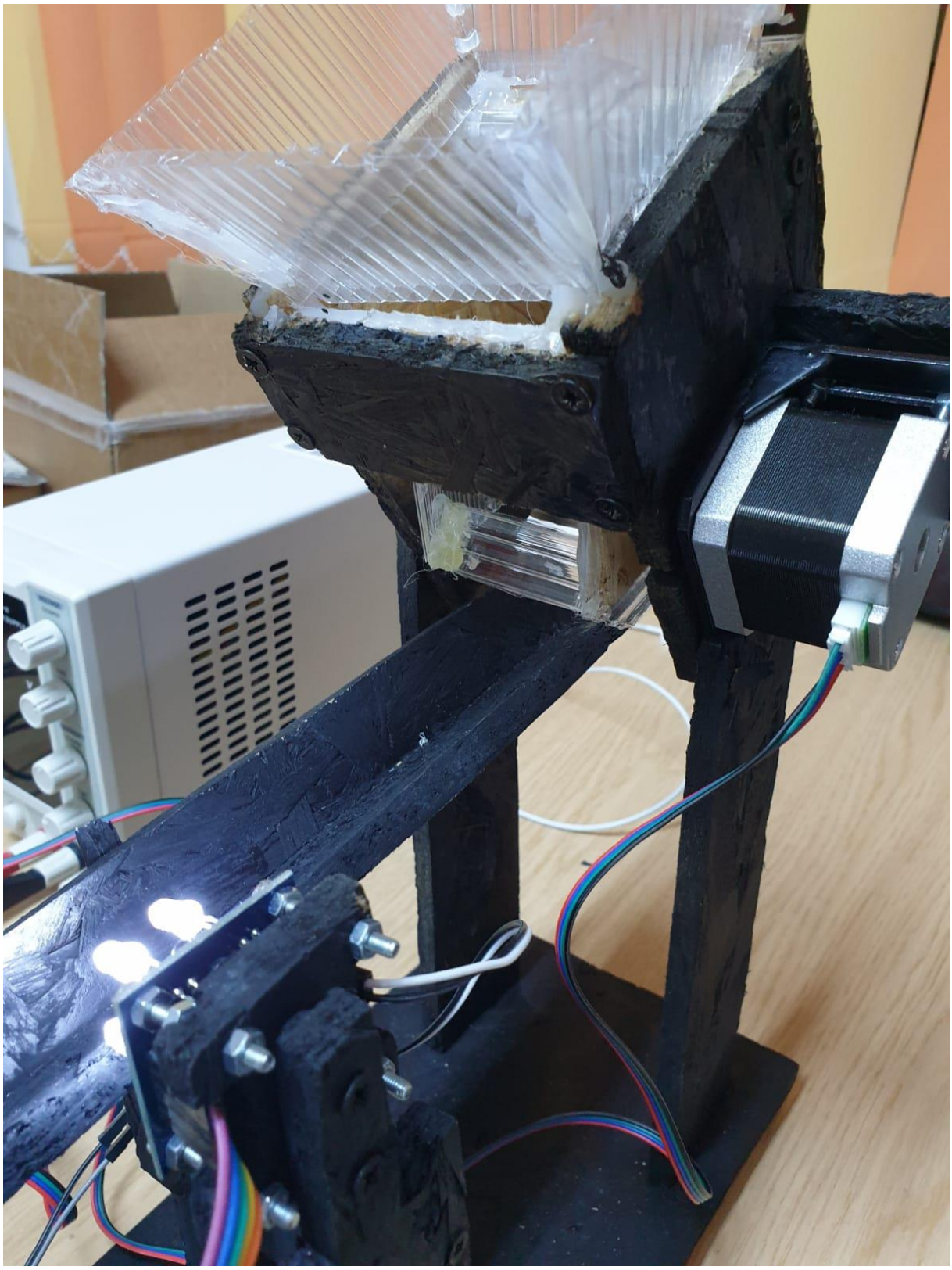
Aici avem fixat motorul care se va ocupa de aruncarea mingilor nepotrivite:

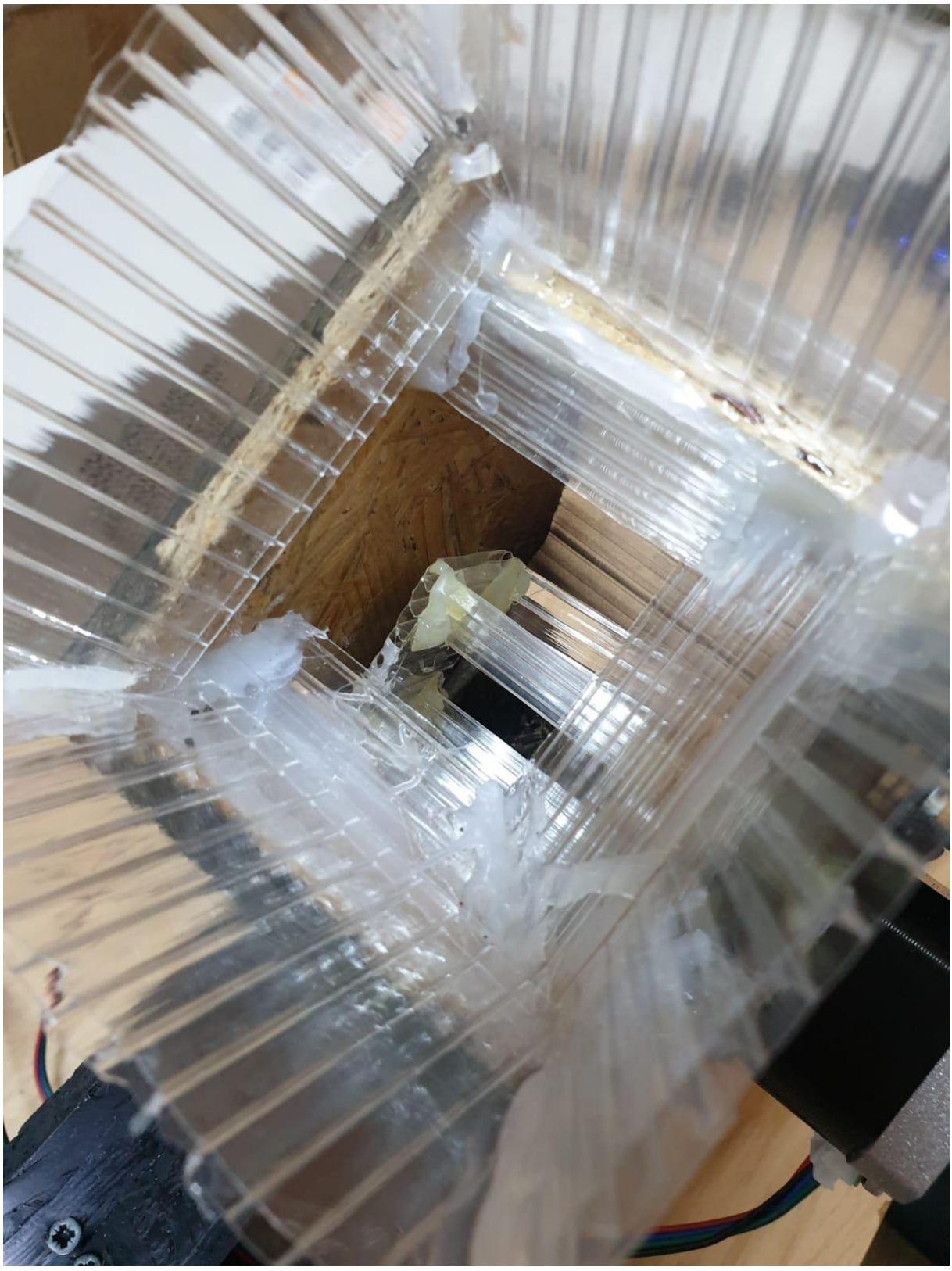


Aici avem fixat motorul care se va de directionarea bilelor in compartimentele configurate in cod, in functie de culoare:

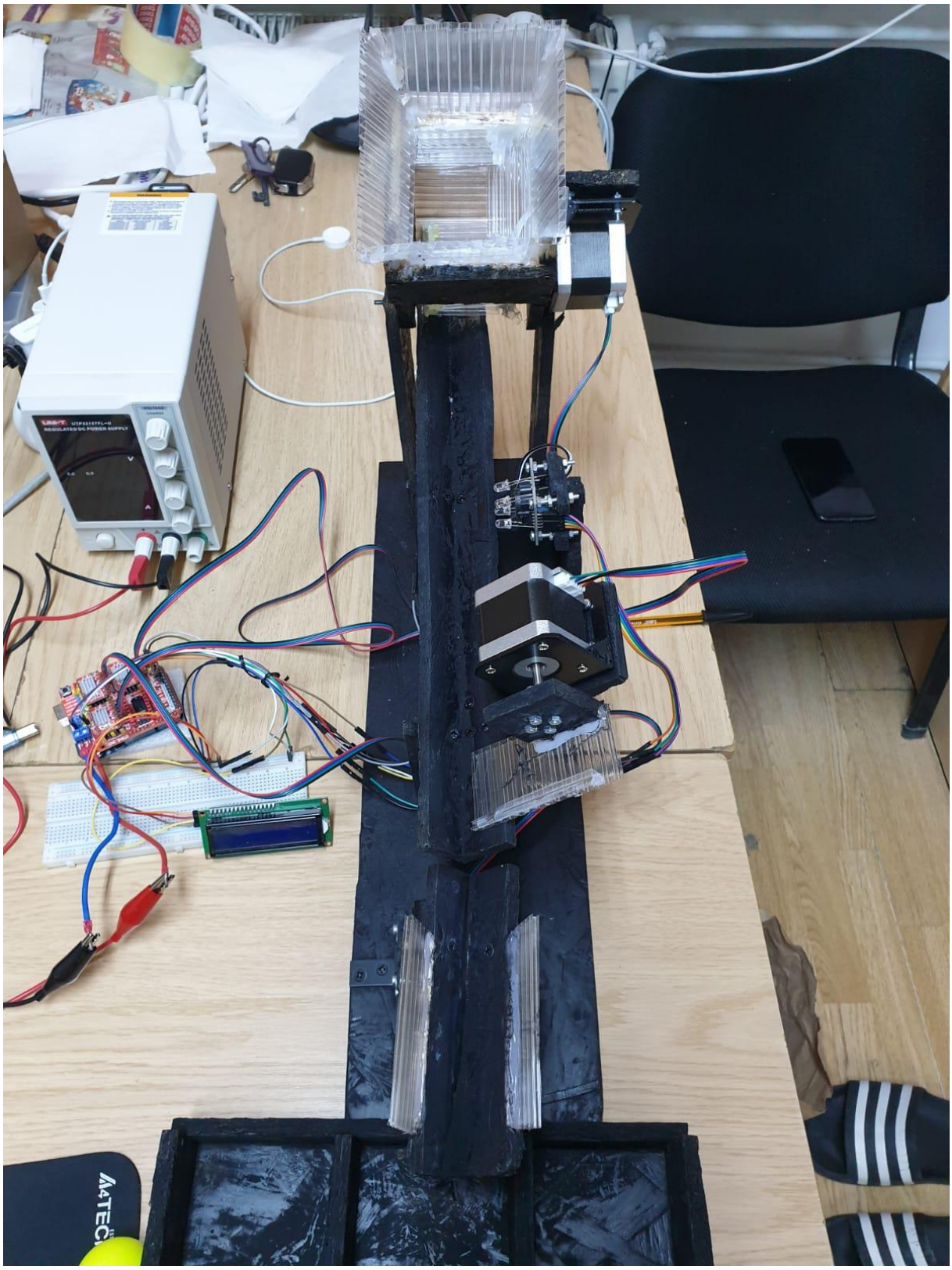


Aici avem fixat motorul care se va ocupa de eliberarea bilelor din palnie, folosind un perete curbat si o inclinatie potrivita pentru a realiza asta:

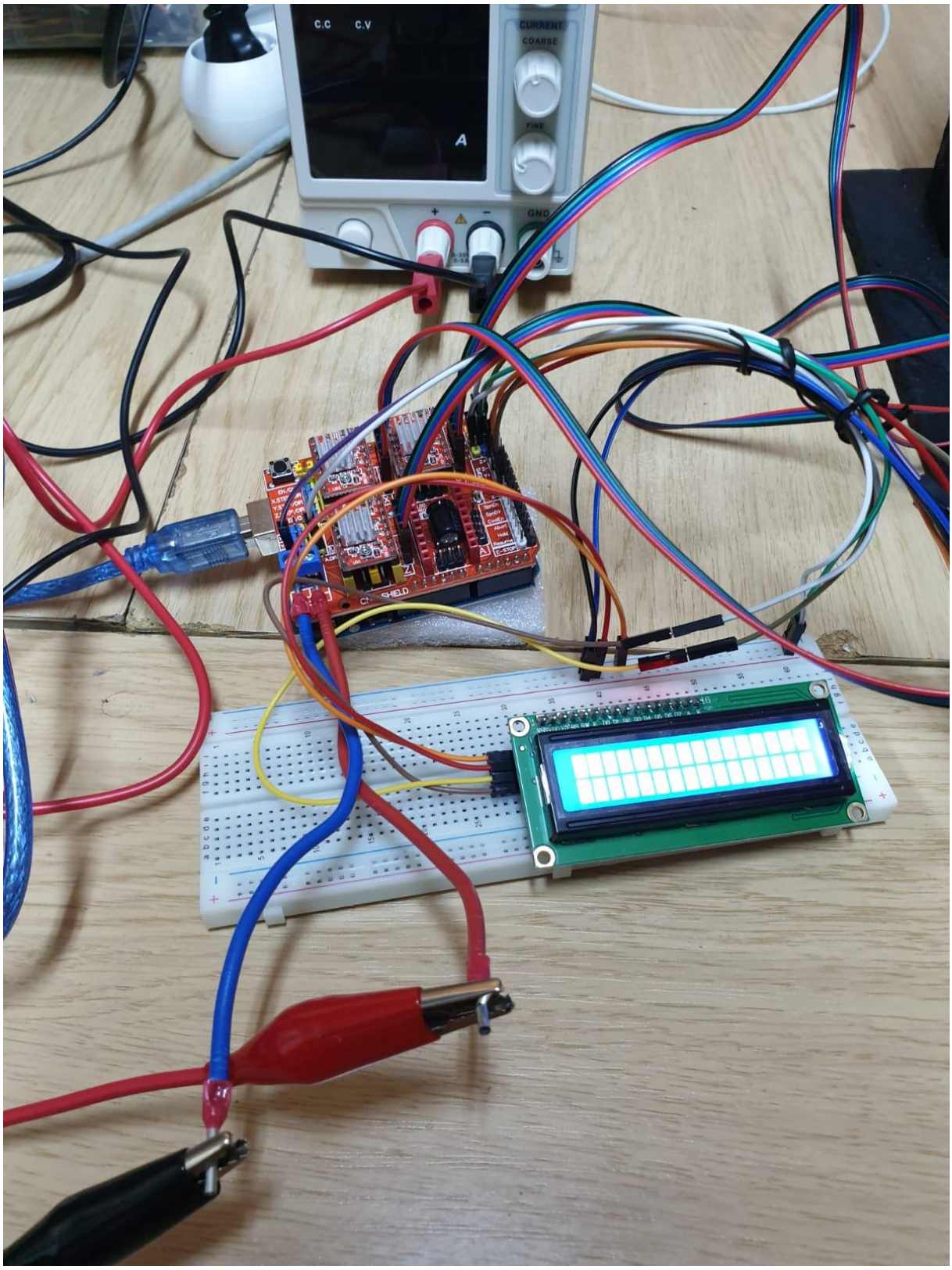




Aici avem macheta finalizata:



- Urmatorul pas este realizarea conexiunilor electrice conform arhitecturii prezentate mai in sus:



Se poate observa ca drivelere sunt echipate cu disipatoare de caldura pentru a nu se supraincalzi.

8.Programul, descrierea codului:

Codul este structurat pe 3 fisiere: 1fisier .c 2 fisiere .h. De asemenea, s-a incercat scrierea cat mai modulara a codului pentru a evita refolosirea de cod si a oferi o intelegere mai usoara a acestuia si pentru o eficienta mai buna.

Fisierul stepper_control.h:

```
#ifndef STEPPER_CONTROL_H
#define STEPPER_CONTROL_H

#define EnableDebuggingLogsStepper true

//cnc shield v3 setup
//arduino pins corresponding to shield pins
const byte StepCS = 2;//x axis
const byte DirCS = 5;
const byte StepTH = 3;//y axis
const byte DirTH = 6;
const byte StepBA = 4;//z axis
const byte DirBA = 7;

//Variables and data structures
const byte DirClockwise = HIGH, DirCounterClockwise = LOW;

struct StepperCscDataVar
{
  const byte LeftContainerPos = 0x00, MiddleContainerPos = 0x01, RightContainerPos = 0x02;
  byte LeftContainerColor = ColorRed, MiddleContainerColor = ColorGreen, RightContainerColor = ColorBlue; //default setup
  byte LeftContainerCnt = 0x00, MiddleContainerCnt = 0x00, RightContainerCnt = 0x00;
  byte CurrentStepperPos = MiddleContainerPos; //Stepper is in 'home' position, pointing the ramp to the middle container
};

struct StepperThDataVar
{
  byte BallThrowsCnt = 0x00;
  bool ThrowRequestActive = false;
};

void MoveStepper(byte StepperPin, int StepsNumber, int StepDelay, byte DirectionPin, byte Direction)
{
  digitalWrite(DirectionPin, Direction);
  StepDelay = (StepDelay < 300) ? 300 : StepDelay; // minimum step delay >=300 to avoid overstepping
  for (int steps = 0; steps < StepsNumber; steps++)
  {
    digitalWrite(StepperPin, HIGH);
    delayMicroseconds(StepDelay);
    digitalWrite(StepperPin, LOW);
    delayMicroseconds(StepDelay);
  }
}
//After using this function, please use resetDataRec() function to clear the data recorder
```

```
void StepperCscContainerWrite(StepperCscDataVar *StepperCSC, byte ContainerLeftColor, byte ContainerMiddleColor, byte ContainerRightColor) {
  StepperCSC->LeftContainerColor  = ContainerLeftColor;
  StepperCSC->MiddleContainerColor = ContainerMiddleColor;
  StepperCSC->RightContainerColor = ContainerRightColor;
}
/*StepperColorSortingControl
  Function to control the the stepper responsible for directing the ball to the required color
*/
void StepperCsMoveHome(StepperCscDataVar *StepperCSC, int StepDelayHome, int StepNo)
{
  if (StepperCSC->CurrentStepperPos != StepperCSC->MiddleContainerPos)
  {
    if (StepperCSC->CurrentStepperPos == StepperCSC->LeftContainerPos)
    {
      MoveStepper(StepCS, StepNo, StepDelayHome, DirCS, DirCounterClockwise);
    } else if (StepperCSC->CurrentStepperPos == StepperCSC->RightContainerPos)
    {
      MoveStepper(StepCS, StepNo, StepDelayHome, DirCS, DirClockwise);
    }
    StepperCSC->CurrentStepperPos = StepperCSC->MiddleContainerPos;
  }
}
void StepperCsControl(byte BallColor, StepperCscDataVar *StepperCSC, int StepNoCs, int StepDelayCs, int PositionHoldDelay)
{
  //StepNo = 25; full step mode corresponding number of steps for 45 degree rotation of the stepper
  //StepDelay=500; // Very high speed is not needed, 500 microseconds delay between steps is enough

  if (StepperCSC->CurrentStepperPos == StepperCSC->MiddleContainerPos)
  {
    if (BallColor == StepperCSC->MiddleContainerColor)
    {
      StepperCSC->MiddleContainerCnt++;
    } else if (BallColor == StepperCSC->LeftContainerColor)
    {
      StepperCSC->CurrentStepperPos = StepperCSC->LeftContainerPos;
      MoveStepper(StepCS, StepNoCs, StepDelayCs, DirCS, DirClockwise);
      StepperCSC->LeftContainerCnt++;
    } else if (BallColor == StepperCSC->RightContainerColor)
    {
      StepperCSC->CurrentStepperPos = StepperCSC->RightContainerPos;
      MoveStepper(StepCS, StepNoCs, StepDelayCs, DirCS, DirCounterClockwise);
      StepperCSC->RightContainerCnt++;
    }
    Serial.print("Color sorted to container: ");
    Serial.println(BallColor);
  }
  delay(PositionHoldDelay);
  StepperCsMoveHome(StepperCSC, StepDelayCs, StepNoCs); //Always moving the stepper in the middle position for simpler logic
}
/*StepperThrowControl
  Function to control the the stepper responsible for throwing the balls with the wrong colors off the ramp
*/
void StepperThControl(int ThrowDelay, int ReturnDelay, int StepNoTH, int StepDelayTh)
{
```

```
  delay(ThrowDelay);/*the ramp incline is constant therefore we asume a constant delay from when the ball is released
  to the moment the throw comand is activated*/
  MoveStepper(StepTH, StepNoTH, StepDelayTh, DirTH, DirClockwise);
  delay(ReturnDelay);//delay for the motor to return to it's initial positon
  MoveStepper(StepTH, StepNoTH, StepDelayTh, DirTH, DirCounterClockwise );
  Serial.println("Th actuator activated");
}
/*StepperBallActuationControl
  Function to control the the stepper responsible for releasing the balls from the funnel to the ramp
*/
void StepperBaControl(int StepDelayBa, int StepNoBa) {
  MoveStepper(StepBA, StepNoBa, StepDelayBa, DirBA, DirCounterClockwise );
}
void StepperCsRun(int color,StepperCscDataVar *StepperCSC,StepperThDataVar *StepperTHC)
{
  if (StepperTHC->ThrowRequestActive == false){
   StepperCsControl(color,StepperCSC, 20, 1400, 4000); //(byte BallColor, StepperCscDataVar StepperCSC, byte StepNoCs, int StepDelayCs, int PositionHoldDelay)
  }
  else{
    StepperTHC->ThrowRequestActive = false;
  }
}
void StepperThRun(byte ball_color, StepperCscDataVar *StepperCSC, StepperThDataVar *StepperTHC)
{
  if ((ball_color != StepperCSC->LeftContainerColor && ball_color != StepperCSC->MiddleContainerColor && ball_color != StepperCSC->RightContainerColor )|| ball_color == ColorUndefined)
  {
    StepperTHC->ThrowRequestActive = true;
    StepperThControl(400, 500, 40, 500); //(int ThrowDelay,int ReturnDelay, byte StepNoTH, byte StepDelayTh)
    StepperTHC->BallThrowsCnt++;
  }
}
void StepperBaRun()
{
  StepperBaControl( 1000, 200);//(int StepDelayBa,int StepNoBa)
}
#endif
```

In acest fisier avem codul pentru controlul motoarelor, avem functia move stepper, care poate deplasa toate motoarele, in functie de parametrii.Pentru unele motoare avem structuri speciale pentru stocarea datelor, care ulterior vor si scrise in strucura globala de inregistrare de date. Pentru motorul numit color select control (CSC) avem functia de moveHome, Control si Run.

- In functia de moveHome, ne asiguram ca motorul este intotdeauna orientat in pozitia de mijloc dupa fiecare operatie pentru a eficientiza codul si celelalte functii,
- In functia control CSC se face logica selectarii containerului potrivit si se contorizeaza datele.
- In functia run sunt centralizati parametrii si este rulat tot codul pentru motorul CSC

Similar aceeasi arhitectura pentru motorul Throw Control si Ball Actuation.
Pentru Th ce este special, este ca asteptam o validare de la senzorul de culoare si facem verificarile corespunzatoare ca sa determinam daca mingea trebuie aruncata sau nu de pe rampa.
Ba este mai simplu, doar se invarte la intervale constante si elibereaza mingile.
Mai avem functia de StepperCscContainerWrite care configureaza culoarea fiecarui container de exemplu rosu galben verde de la stanga la dreapta.

Fisierul color_detection.h:

```
#ifndef COLOR_DETECTION_H
#define COLOR_DETECTION_H
#define S0 12
#define S1 13
#define S2 10
#define S3 11
#define FREQ_OUT 9
#define EnableDebuggingLogsColors false
const byte ColorUndefined = 0, ColorRed = 1, ColorGreen = 2, ColorBlue = 3, ColorYellow = 4,
ColorWhite = 5;
const byte MAX_COLORS = 6;
int redFrequency = 0;
int greenFrequency = 0;
int blueFrequency = 0;
int colorBlueDetect = 0;
int colorRedDetect = 0;
int colorGreenDetect = 0;
int freq_scan_delay = 0;
byte BallColor = ColorUndefined, BallColorPrev = ColorUndefined, BallColorOut =
ColorUndefined;
int sample_buffer [5], color_app[MAX_COLORS];
int sample_mean_crr = 0, sample_mean_prev = 0;
byte i;
bool ball_detected = false;
//circular buffer sampling
const byte window_sampling_size = 5;
int R_buffer[window_sampling_size];
int G_buffer[window_sampling_size];
int B_buffer[window_sampling_size];
int buff_index = 0;
float R_avg_prev = 0.0, G_avg_prev = 0.0, B_avg_prev = 0.0;
const float movement_detect_tresh = 40.0;
const byte colorDetectOffset=50;


void init_buffers() {
  for (i = 0; i < window_sampling_size; i++) {
    R_buffer[i] = 0;
    G_buffer[i] = 0;
    B_buffer[i] = 0;
  }
}
void setup_color_detect(int freq_delay) {
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  pinMode(FREQ_OUT, INPUT);
  digitalWrite(S0, HIGH);
  digitalWrite(S1, LOW);
```

```
  freq_scan_delay = freq_delay;
  init_buffers();
}
void read_frequency()
{
  // Setting up the photodiodes for each color and reading the frequency
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  redFrequency = pulseIn(FREQ_OUT, LOW);
  delay(freq_scan_delay);
  digitalWrite(S2, HIGH);
  digitalWrite(S3, HIGH);
  greenFrequency = pulseIn(FREQ_OUT, LOW);
  delay(freq_scan_delay);
  digitalWrite(S2, LOW);
  digitalWrite(S3, HIGH);
  blueFrequency = pulseIn(FREQ_OUT, LOW);

#if EnableDebuggingLogsColors == true
  Serial.println("Color detected without filtering function:");
  Serial.print("R = ");
  Serial.print(redFrequency);
  Serial.print(" G = ");
  Serial.print(greenFrequency);
  Serial.print(" B = ");
  Serial.println(blueFrequency);
#endif

  delay(freq_scan_delay);
}
void color_detect_routine() {
  byte color_max_cnt = 0, color_max_index = 0;
  int R_sum_window = 0, G_sum_window = 0, B_sum_window = 0;
  float R_avg, G_avg, B_avg;
  float R_diff, G_diff, B_diff;
  bool sampling_active = false, output_color_valid = false;
  byte sample_count = 0, valid_cnt = 0;
  ball_detected = false;
  if (ball_detected != true)
    Serial.println("Ball detection routine active!");
  while (ball_detected != true) {
    read_frequency();
    colorBlueDetect = map(blueFrequency, 10, 600, 1023, 0);
    colorRedDetect = map(redFrequency, 10, 600, 1023, 0);
    colorGreenDetect = map(greenFrequency, 10, 600, 1023, 0);
    //buffer operation start
    R_buffer[buff_index] = colorRedDetect;
    G_buffer[buff_index] = colorGreenDetect;
    B_buffer[buff_index] = colorBlueDetect;

    buff_index = (buff_index + 1) % window_sampling_size;

    R_sum_window = 0;
```

```
    G_sum_window = 0;
    B_sum_window = 0;

    for (i = 0; i < window_sampling_size; i++) {
      R_sum_window += R_buffer[i];
      G_sum_window += G_buffer[i];
      B_sum_window += B_buffer[i];
    }

    R_avg = (float)R_sum_window / window_sampling_size;
    G_avg = (float)G_sum_window / window_sampling_size;
    B_avg = (float)B_sum_window / window_sampling_size;

    R_diff = abs(R_avg - R_avg_prev);
    G_diff = abs(G_avg - G_avg_prev);
    B_diff = abs(B_avg - B_avg_prev);

  #if EnableDebuggingLogsColors == true
    Serial.println("Colors avg, avg_prev difference:");
    Serial.print("R_diff = ");
    Serial.print(R_diff);
    Serial.print(" G_diff = ");
    Serial.print(G_diff);
    Serial.print(" B_diff = ");
    Serial.println(B_diff);
    Serial.print("R_avg = ");
    Serial.print(R_avg);
    Serial.print(" G_avg = ");
    Serial.print(G_avg);
    Serial.print(" B_avg = ");
    Serial.println(B_avg);
    Serial.print("R_avg_prev = ");
    Serial.print(R_avg_prev);
    Serial.print(" G_avg_prev = ");
    Serial.print(G_avg_prev);
    Serial.print(" B_avg_prev = ");
    Serial.println(B_avg_prev);
    Serial.print(" Threshhold = ");
    Serial.println(movement_detect_tresh);
    Serial.println();

  #endif
  #if EnableDebuggingLogsColors == true
    Serial.println("Color detected with filter function:");
    Serial.print("R = ");
    Serial.print(colorRedDetect);
    Serial.print(" G = ");
    Serial.print(colorGreenDetect);
    Serial.print(" B = ");
    Serial.println(colorBlueDetect);
  #endif
    sample_mean_crr = (colorGreenDetect + colorRedDetect + colorBlueDetect) / 3.0;
  #if EnableDebuggingLogsColors == true
```

```
   Serial.print(sample_mean_crr);
   Serial.print(" > Sample prevf-: ");
   Serial.print(sample_mean_prev - 50);
   Serial.print(" Sample crr: ");
   Serial.print(sample_mean_crr);
   Serial.print(" < Sample prevf+: ");
   Serial.println(sample_mean_prev + 50);
#endif

   if (output_color_valid == true) {
     // Serial.print("Sampling_active_flag: ");
     //  Serial.println(sampling_active);

     // if ( (sample_mean_crr > sample_mean_prev - 100 ) && (sample_mean_crr < sample_mean_prev + 100)) // idle -- 200 prev
     if ((R_diff < movement_detect_tresh && G_diff < movement_detect_tresh && B_diff < movement_detect_tresh) && sampling_active == false)
     {
#if EnableDebuggingLogsColors == true
       Serial.println("No movement detected");
#endif
     } else // movement detected
     {
#if EnableDebuggingLogsColors == true
       Serial.println("Movement detected, treshold condition active");
       if (sampling_active == false)
         Serial.println("Initiating sampling routine.");
       Serial.print("White condition: ");
       Serial.println((colorRedDetect + colorBlueDetect + colorGreenDetect) / 3);
#endif
       BallColor = ColorUndefined;
       if (colorRedDetect > colorBlueDetect + colorDetectOffset && colorGreenDetect > colorBlueDetect + colorDetectOffset && (abs(colorRedDetect - colorGreenDetect) <= 30)) //20 previous
         BallColor = ColorYellow;
       else
       {
         if (colorRedDetect > colorGreenDetect+colorDetectOffset && colorRedDetect > colorBlueDetect+colorDetectOffset)
           BallColor = ColorRed;
         else if ( (float)( colorRedDetect + colorGreenDetect + colorBlueDetect) / 3 > 920) //920 threshold for white ball, we use it as undefined in tests for throwing
           BallColor = ColorWhite;
         else if (colorGreenDetect > colorRedDetect && colorGreenDetect > colorBlueDetect)
           BallColor = ColorGreen;
         else if (colorBlueDetect > colorRedDetect && colorBlueDetect > colorGreenDetect)
           BallColor = ColorBlue;
         else
           BallColor = ColorUndefined;
       }
#if EnableDebuggingLogsColors == true
       Serial.print("Detected ball color is: ");
       Serial.println(BallColor);
```

```
#endif
      //if(BallColor!=BallColorPrev && sampling_active == false){
      if (sampling_active == false) {
        Serial.println("Sampling sequence initiated");
        sampling_active = true;
        sample_count = 0;
        color_max_cnt = 0;
        color_max_index = 0;
      }
      if (sampling_active == true)
      {
        sample_count++;
        if ( sample_count < 12)//15 is good also
        {
          color_app[BallColor]++;
        }
        else
        {
          for (byte colors_index = 0; colors_index < MAX_COLORS; colors_index++)
            if (color_app[colors_index] > color_max_cnt)
            {
              color_max_cnt = color_app[colors_index];
              color_max_index = colors_index;
            }
#if EnableDebuggingLogsColors == false
          Serial.print("MaxC: ");
          Serial.print(color_max_index);
          Serial.print(" Color counter for sampling cycle: ");
#endif

          for (i = 0; i < MAX_COLORS; i++) {
#if EnableDebuggingLogsColors == false
            Serial.print(color_app[i]);
            Serial.print(" ");
#endif
            color_app[i] = 0;
          }
#if EnableDebuggingLogsColors == false
          Serial.println();
#endif

          BallColorOut = color_max_index;
          ball_detected = true;

          sampling_active = false;
          Serial.println("Sampling sequence completed");
#if EnableDebuggingLogsColors == false
          Serial.print("Out ball color is: ");
          Serial.println(BallColorOut);
#endif
        }
      }
      BallColorPrev = BallColor;
```

```
    }
   }
   sample_mean_prev = sample_mean_crr;
   R_avg_prev = R_avg;
   G_avg_prev = G_avg;
   B_avg_prev = B_avg;

   if (valid_cnt < 7) // validate color sampling routine after 7 reading cycles to stabilise the output
    valid_cnt++;
   else
    output_color_valid = true;
  }


  Serial.println("Ball detected!");
  // delay(100);
}
#endif
```

Aici descrierea pe scurt, ar fi ca avem o functie care citeste frecventele de la fiecare ansamblu de fotodiode, dupa care datele sunt prelucrate in functia color_detect_routine(). Sunt amplasate compile switch-uri in cod pentru vizualizarea datelor, depanare.
In functia de detectare a culorilor, se incarca frecventele mapate pentru fiecare din cele 3 ansambluri de fotodiode: rosu verde albastru in un buffer circular, care va memora ultimele 5 cicluri de citire si va face media. Aceasta medie va fi comparata cu media precedenta pentru a detecta cand un obiect apare in fata senzorului de culoare. Putem detecta acest lucru deoarece diferenta intre lumina ambientala si lumina unui obiect aproape de senzor e mare.
Deci putem face diferenta intre cele 2 medii si cu un prag putem determina daca avem miscare sau nu.
Daca avem miscare, incepem un ciclu de esantionare pentru a detecta cu acuratete mare culoarea obiectului. S-a determinat experimental ca 12 esantioane cu intarziere de 5 milisecunde intre citirile fotodiodelor sunt potrivite pentru a spune cu precizie culoarea, avand in vedere ca mingea coboara cu viteza pe rampa. Se vor lua 12 esantioane consecutive ale citirilor curente ale senzorului si se vor introduce culorile citite in un vector de aparitii corespunzator culorilor, culoarea cu cele mai multe aparitii va fi transmisa la iesire. Acest pas este necesar datorita luminozitatii ambientale si reflexiei luminii si vitezei mingii pot aparea citiri eronate, cu acest procedeu rezolvam aceasta problema.

Fisierul project_color_sorting_main.c

```
#include <stdlib.h>
#include<LiquidCrystal_I2C.h>
#include<Wire.h>
#include "color_detection.h"
#include "stepper_control.h"

//module compile switches
#define EnableDebuggingLogsMain true

//Variables and data structures
```

```
LiquidCrystal_I2C lcd(0x27, 16,2);
byte Reset = 0;
String str;

struct DataRecorderVar
{
  byte LeftContainerCnt = 0x00, MiddleContainerCnt = 0x00, RightContainerCnt = 0x00;
  byte BallThrowsCnt = 0x00;
};
DataRecorderVar DataRecorder;

StepperCscDataVar StepperCscData;
StepperThDataVar StepperThData;

//Functions
void PinSetup() {
  pinMode(StepCS, OUTPUT);
  pinMode(DirCS , OUTPUT);
  pinMode(StepTH, OUTPUT);
  pinMode(DirTH , OUTPUT);
  pinMode(StepBA, OUTPUT);
  pinMode(DirBA , OUTPUT);
}
void WriteStepperDataToDataRec(StepperCscDataVar *StepperCSC, StepperThDataVar
*StepperTh, DataRecorderVar *DataRec)
{
  DataRec->LeftContainerCnt = StepperCSC->LeftContainerCnt;
  DataRec->MiddleContainerCnt = StepperCSC->MiddleContainerCnt;
  DataRec->RightContainerCnt = StepperCSC->RightContainerCnt;
  DataRec->BallThrowsCnt = StepperTh->BallThrowsCnt;
}
void printDataRecToSerial(){
  Serial.println("Data Recorder Data:");
  Serial.print("LeftContainerCnt:");
  Serial.print(DataRecorder.LeftContainerCnt);
  Serial.print(" MiddleContainerCnt:");
  Serial.print(DataRecorder.MiddleContainerCnt);
  Serial.print(" RightContainerCnt:");
  Serial.print(DataRecorder.RightContainerCnt);
  Serial.print(" BallThrowsCnt:");
  Serial.println(DataRecorder.BallThrowsCnt);
  Serial.println();
}
void resetDataRec(){
  DataRecorder.LeftContainerCnt=0;
  DataRecorder.MiddleContainerCnt=0;
  DataRecorder.RightContainerCnt=0;
  DataRecorder.BallThrowsCnt=0;
}

void printDataRecToLCD(){
    lcd.clear();
    lcd.setCursor(0,0);
```

```
   lcd.print("Nr. colrs sorted");
   lcd.setCursor(0,1);
   str = "R:" + String(DataRecorder.LeftContainerCnt) + "
G:"+String(DataRecorder.MiddleContainerCnt)+ "
B:"+String(DataRecorder.RightContainerCnt)+" T:"+String(DataRecorder.BallThrowsCnt);
   lcd.print(str);
}
void lcd_scrolling_display( char *str_buffer_row0,char *str_buffer_row1){
  int offset0,offset1;
  char lcd_row[16];

for(offset0=0,offset1=0;offset0<strlen(str_buffer_row0),offset1<strlen(str_buffer_row1);offset0++
,offset1++){
strncpy(lcd_row , str_buffer_row0+offset0, 16);
lcd.setCursor(0,0);
lcd.print(lcd_row);
strncpy(lcd_row , str_buffer_row1+offset1, 16);
lcd.setCursor(0,1);
lcd.print(lcd_row);
delay(100);
lcd.clear();
delay(100);
  }
}
void setup() {
  Serial.begin(115000);
  PinSetup();
  lcd.init();
  lcd.backlight();
  //Initial display message
  lcd_scrolling_display("     Ball Color Sorting System          "," By Kafka Patrik, year 3 UPT
CTI RO 2023");
  lcd.setCursor(0,0);
  lcd.print("Waiting for move");
  lcd.setCursor(0,1);
  lcd.print("ment....");
  setup_color_detect(5);//10 before, delay 5 miliseconds between each photodiode array read from
color sensor
  StepperCscContainerWrite(&StepperCscData, ColorRed, ColorYellow, ColorBlue); //Default
value
  StepperCscData.CurrentStepperPos = StepperCscData.MiddleContainerPos;//initial position
  delay(5000);
}

void loop() {
  StepperBaRun();
  color_detect_routine();
  StepperThRun(BallColorOut, &StepperCscData, &StepperThData);
  StepperCsRun(BallColorOut,&StepperCscData, &StepperThData);
  WriteStepperDataToDataRec(&StepperCscData, &StepperThData, &DataRecorder);
  //printDataRecToSerial();
  printDataRecToLCD();
  delay(200);
```

}

Aici sunt declarate functiile si variabilele pentru structura de inregistrare globala si interfetele pentru afisarea datelor pe LCD, sau monitorul serial. Aici se fac configuratiile initiale si se ruleaza programul principal.

1. Intai se activiaza motorul Ba pentru a elibera o bila
2. Apoi se activeaza rutina de detectie a culorii, care asteapta bila si apoi detecteaza culoare si transmite un semnal la Th
3. Daca semnalul la Th este pozitiv inseamna ca bila trebuie aruncata de pe rampa
4. Altfel se va face directionarea in compartimente pe culori cu motorul CSC
5. Apoi se vor scrie datele din structurile in structura de inregistrare globala a datelor.
6. Apoi se vor afisa prin diferite interfete pe dispozitivele de afisare cerute: LCD, monitor serial.
7. Apoi bucla se repeta si coboara urmatoarea minge.

9.Date experimentale:

Am testat 10 aruncari cu urmatoarele bile: Rosu,Galben,Albastru,Alb.

Bilele rosii si albastre au fost citite corect de fiecare data.

Bilele albe si galbene au fost detectate corect de 9 respectiv 8 ori din 10. Acest lucru se intampla din cauza culorii apropiate a mingii albe si verde – deschis pentru ca mediile lor sunt foarte apropiate.

Aceasta problema se poate rezolva prin adaptarea pragurilor pentru culoare pentru luminozitatea curenta.

10.Estimare cost/timp:

Timpul pentru asamblarea machetei aproximativ 20 de ore.

Timpul pentru scrierea codului in jur de 12-14 ore.

Costul aproximativ la momentul acesta s-ar ridica la:

1. Costul placajului, policarbonatului si suruburilor aproximativ 30 de lei.
2. Costul motoarelor nema 17 46 lei *3 = 138 lei
3. Cost suporturi motoare 7*3= 21 lei
4. Hub-uri aluminiu, pentru fixare pe tija motorului set 4 buc 80 lei
5. Cost drivere 4*11 lei =44 lei
6. Cost Shield cnc = 25 lei

7. Senzor culoare 40 lei
8. Modul lcd cu i2c 16x2 =25 lei
9. Plusivo Uno R3 + cablu = 40 lei
10. BreadBoard + cabluri aproximativ 40 lei
11. Optional sursa regulata tensiune DC 30V 5A: 300 lei

Cost total:453 lei + 300 lei sursa

Timp total: 32-34 ore.

11.Imbunatatiri posibile, concluzii

- Implementare sisteme reduntanta.
- Inregistrare a mai multor date.
- Adaugare modul bluetooth, aplicatie mobila.
- Reset global prin buton extern, sau cod pentru resetarea structurilor de date.
- Imbunatatire sistem de eliberare al mingilor.
- Adaugare detectia culorilor compuse.

Demonstratie practica:

WhatsApp Video 2023-05-21 at 23.57.28.mp4

- 12.Surse:

- https://s3-sa-east-1.amazonaws.com/robocore-lojavirtual/889/TCS230%20Datasheet.pdf
- https://lastminuteengineers.com/tcs230-tcs3200-color-sensor-arduino-tutorial/
- https://datasheetspdf.com/pdf-file/1260602/Schneider/NEMA17/1
- https://components101.com/sites/default/files/component_datasheet/A4988%20Stepper%20Motor%20Driver%20Module.pdf
- https://ardufocus.com/howto/a4988-motor-current-tuning/
- https://www.e-jpc.com/stepper-motor-voltage-explained/
- https://circuitdigest.com/microcontroller-projects/controlling-nema-17-stepper-motor-with-arduino-and-a4988-stepper-driver-module

- https://www.youtube.com/watch?v=JlhjcTh4yts - Arduino CNC shield overview
- https://www.researchgate.net/figure/CNC-shield-CNC-shield-V3-is-an-open-source-hardware-used-to-control-stepper-motors_fig2_344000935
- https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/
- https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/
- https://profs.info.uaic.ro/~arduino/index.php/Comunicare_I2C
- https://mikroelectron.com/Product/ARDUINO-CNC-SHIELD-V3/
- https://ro.wikipedia.org/wiki/Atmega328
- https://www.seeedstudio.com/blog/2019/10/22/atmega328p-the-one-microcontroller-you-should-start-with/
- https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
- https://cleste.ro/arduino-nano-v3.html
- https://cleste.ro/arduino-uno-r3-atmega328p.html