Universitatea
Transilvania
din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR

Proiect Calculatoare SI An IV

Închidere centralizată mașină (Proiect Arduino)

ARDUINO

Student: Momoi Alexandru

Mail student: alexandru.momoi@student.unitbv.ro

Grupa: 4LF792 CALCULATOARE

An de studiu: IV

Cuprins

1.Introducere

Scopul general al sistemului

Proiectul are ca scop crearea unui sistem ce se bazează pe închiderea centralizată a unei mașini din telecomandă. La apăsarea butoanelor respective de pe telecomandă, ușile mașinii vor fi încuiate, dublu încuiate sau descuiate, iar în cazul în care geamurile sunt coborâte, acestea se vor ridica. Stările respective vor fi semnalizate prin clipirea ledului pentru un număr de secunde.

În plus apăsarea butonului și starea curentă va fi afișată pe ecranul LDC.

Consider, că proiectul este util mai ales în domeniul automotivelor. Am ales această tema de proiect, întrucât am fost interesat de domeniul auto și de mediul de dezvoltare Arduino.

Stadiul actual al domeniului ales

Pe parcursul anilor, închiderea centralizată la mașini a devenit tot mai performantă. Astfel, în prezent dispunde de funcții precum comanda de la distanță, închiderea și deschiderea ușilor, a geamurilor și a portbagajului.

Niște funcții mai noi sunt cele de START/STOP, parcare mașină (deplasarea mașinii în față/spate), pornire încălzire, pornire faruri, pornire alarmă (prevenție anti-furt), descuiere automată la apropierea de automobil dar și alte îmbunătățiri prin cheia digitală.

În prezent, există și o alternativă la cheie/telecomandă și anume cheia digitală prin smartphone sau smartwatch, unde se realizează aceleași funcții ca prin cheia clasică, doar că la un model mai avansat și modern. Cheia digitală oferă o securitate mai ridicată datorită tehnologiilor moderne de criptare în automobil și pe smartphone.

2. Descriere mediu de lucru Arduino

Arduino este o platformă open-source folosită pentru construirea de proiecte electronice. Arduino constă atât dintr-o placă de circuit fizic programabilă (denumită adesea

microcontroler) cât și dintr-o bucată de software sau IDE (Integrated Development Environment) care rulează pe computer, folosit pentru a scrie și a încărca codul computerului pe placa fizică**.**

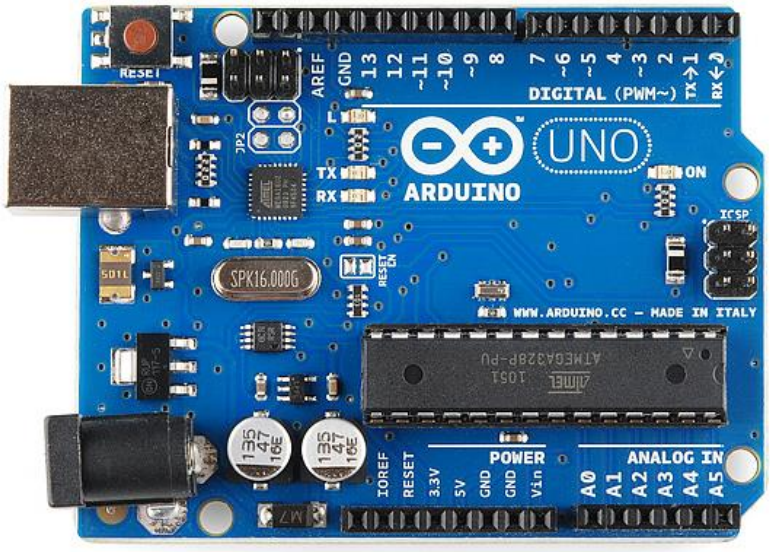
Platforma Arduino a devenit destul de populară în rândul persoanelor care încep cu electronica și din motive întemeiate. Spre deosebire de majoritatea plăcilor de circuite programabile anterioare, Arduino nu are nevoie de o piesă hardware separată (numită programator) pentru a încărca cod nou pe placă - puteți utiliza pur și simplu un cablu USB. În plus, Arduino IDE folosește o versiune simplificată de C++, ceea ce face mai ușor de învățat să programați. În cele din urmă, Arduino oferă un factor de formă standard care descompune funcțiile micro-controlerului într-un pachet mai accesibil.

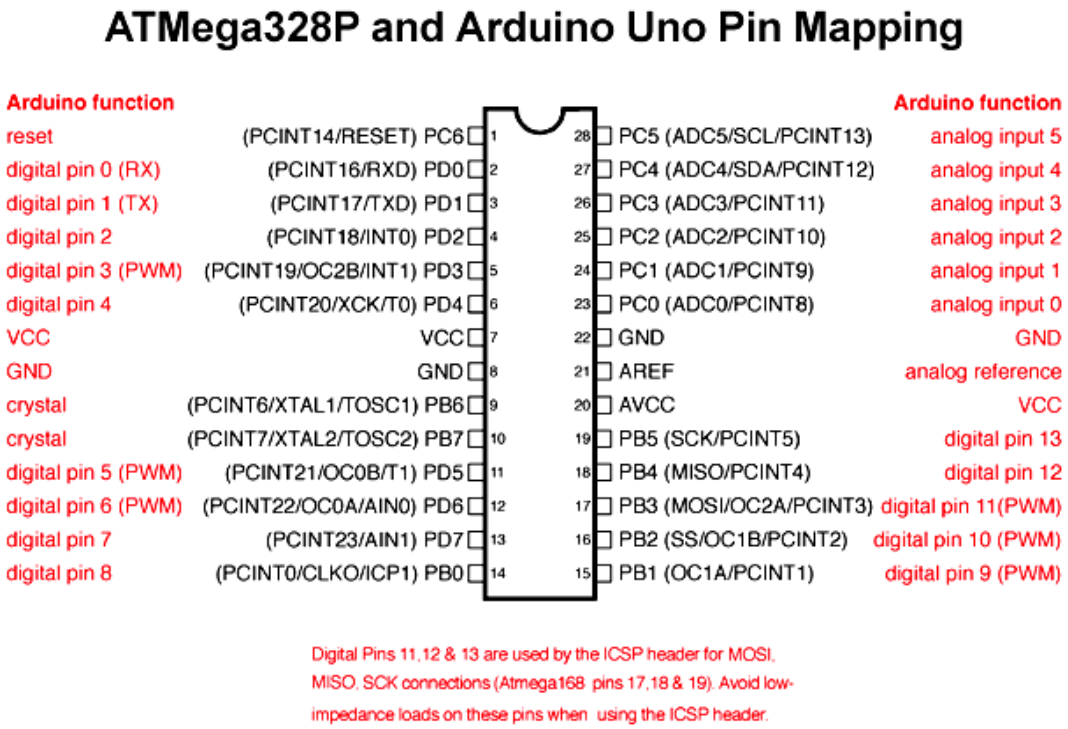
Hardware-ul și software-ul Arduino au fost concepute pentru artiști, designeri, pasionați, hackeri, începători și oricine este interesat să creeze obiecte sau medii

interactive. Arduino poate interacționa cu butoane, LED-uri, motoare, difuzoare, unități GPS, camere, internet și chiar și telefonul inteligent sau televizorul tău!

Arduino Uno este cea mai populară placă din familia Arduino. Este alimentat de cipul ATmega328p, care are 32K de octeți de memorie de program Flash, 2k de octeți de SRAM și 1K de octeți de EEPROM.

Fig 1. Placă Arduino UNO





2.3. Pinii de pe placă

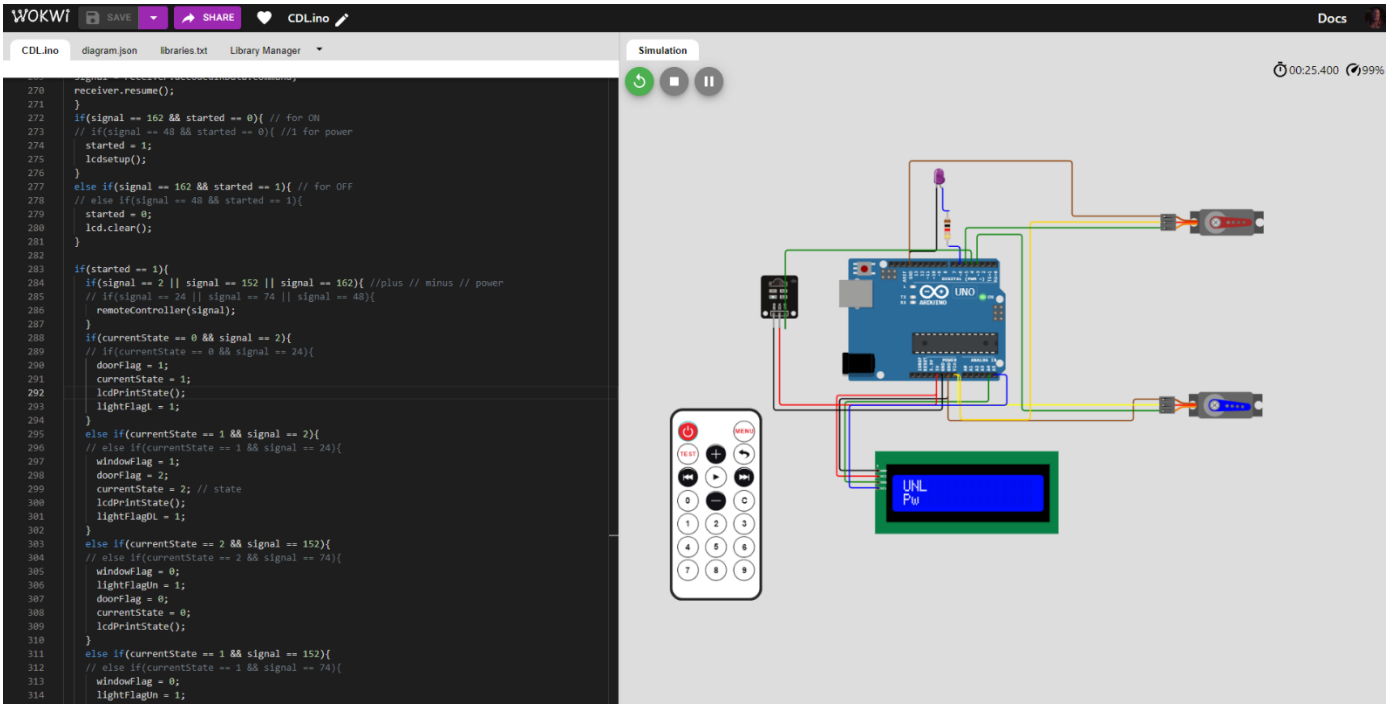
- Pinii de la 0 la 13 sunt pini GPIO digitali. Pinii de la A0 la A5 dublează ca pini de intrare analogică, pe lângă faptul că sunt pini GPIO digitali.
- Există trei pini de masă: GND.1, care se află în partea de sus a plăcii, lângă pinul 13, și GND.2/GND.3, care se află în partea de jos.
- Pinii VIN / 5V sunt conectați la sursa de alimentare pozitivă.
- Pinii 3.3V / IOREF / AREF / RESET nu sunt disponibili în simulare.
- Pinii digitali 3, 5, 6, 9, 10 și 11 au suport hardware PWM.

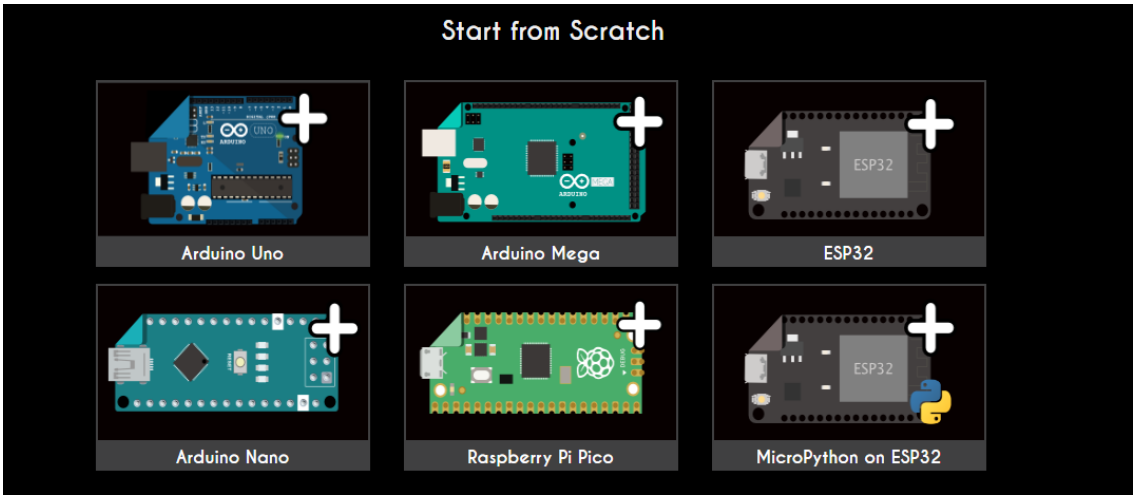
2.1. Descriere mediu de lucru Wokwi

Wokwi este un simulator de electronice online ,pe browser . Se poate utiliza , făcând simulări pe plăcuțe Arduino, ESP32 , Raspberry Pi și multe alte plăci populare, componente și senzori.

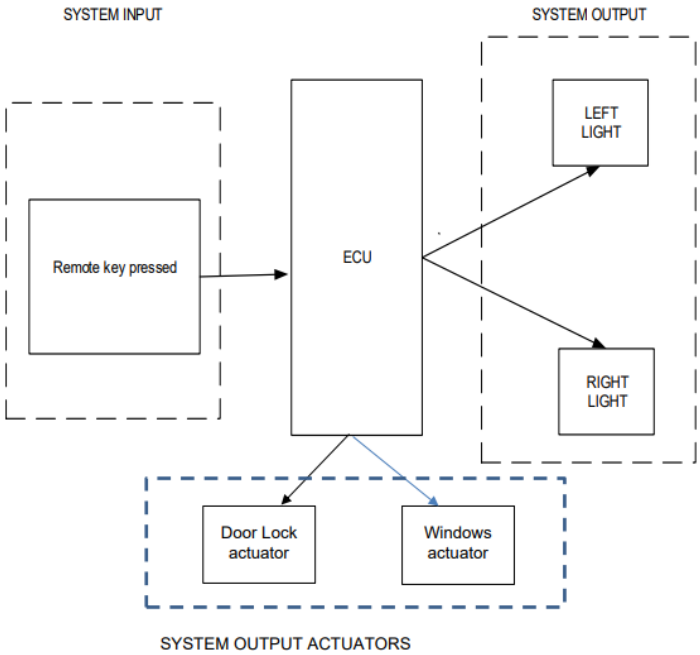
Este util întrucât nu trebuie descărcate software-uri imense, nu există un cost de componente deoarece site-ul dispune de toate perifericele hardware disponibile în librăriile incorporate.

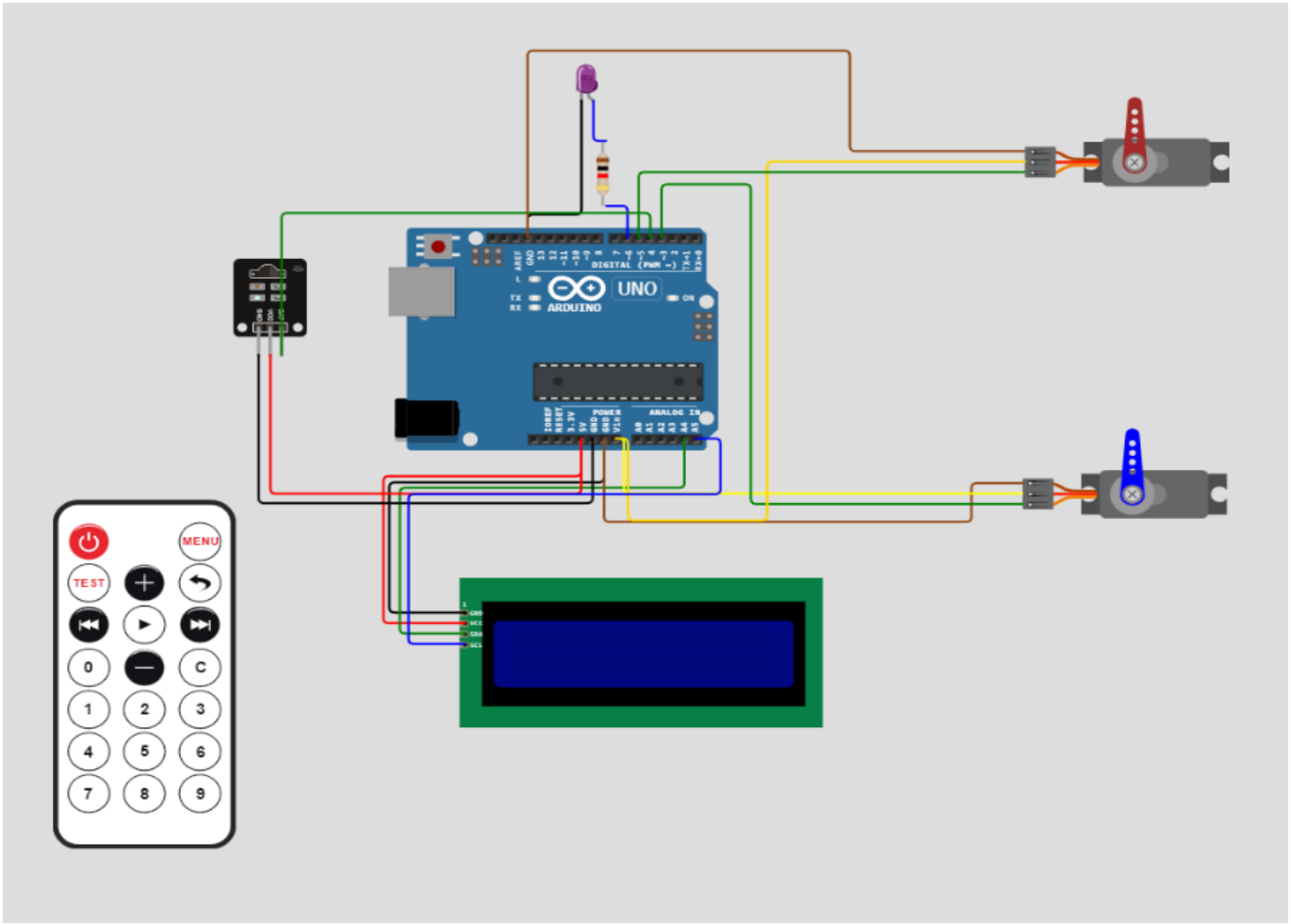
Alte beneficiu imens îl reprezintă riscul scăzut de a strica componentele hardware virtuale, ceea ce incurajează experimentele pe plăcuță fără frica de a produce scurt-circuitarea perifericelor.





3. Arhitectura sistemului – Schema bloc, electrică și montajul propriu zis.



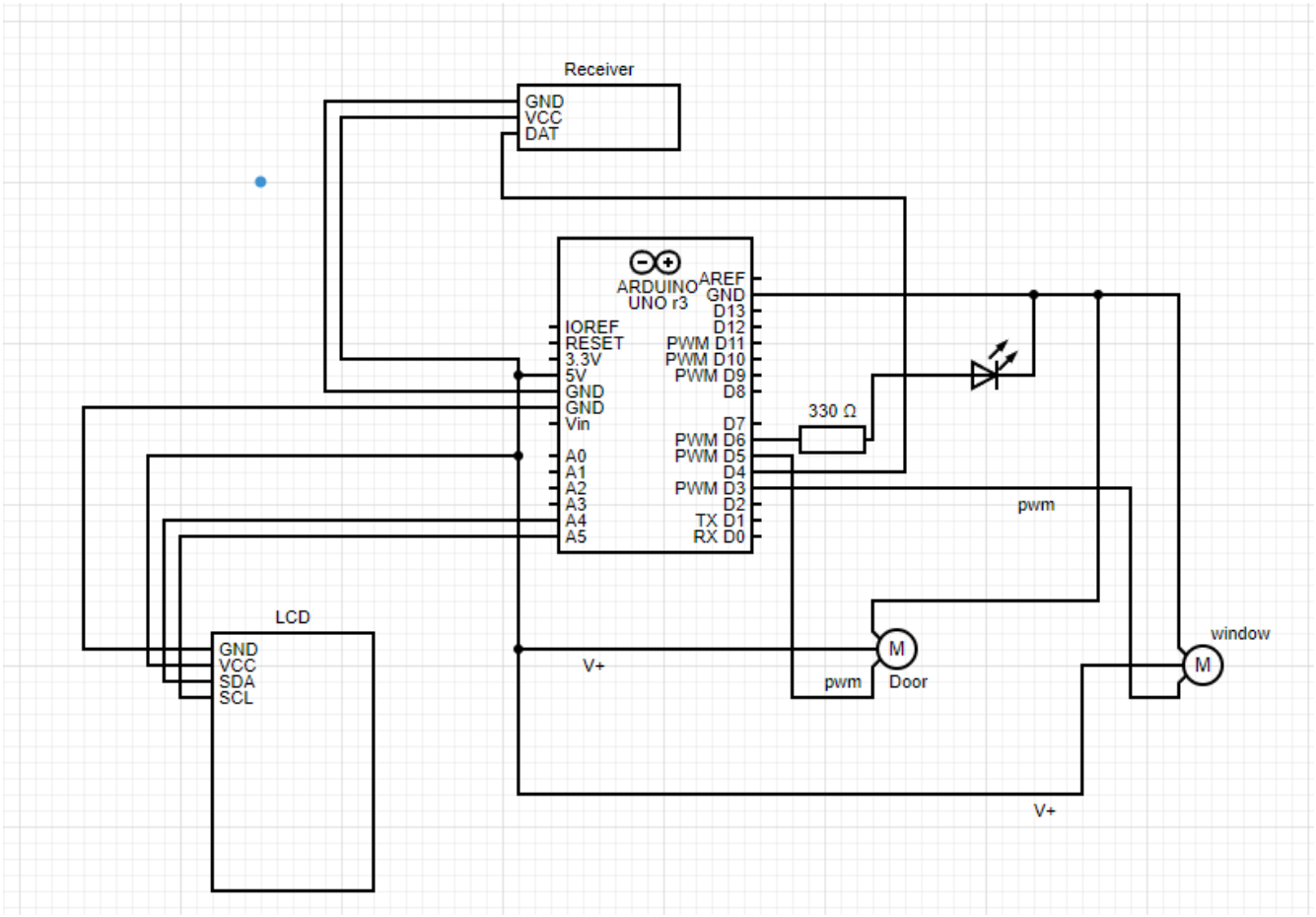


Schema bloc prezintă funcționarea sistemului și perifericele de input și output:

-Telecomanda reprezintă perifericul de tip input

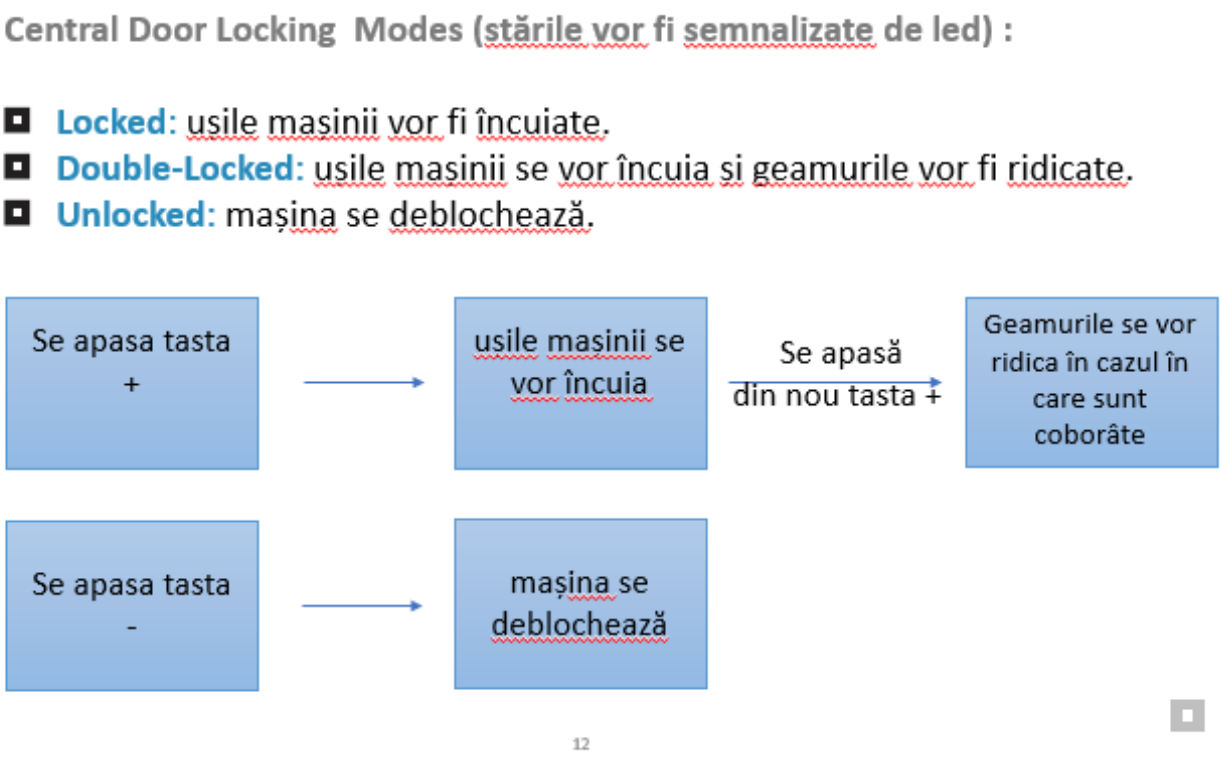
-Servomotoarele (Cele 2) reprezintă perifericele de tip output alături de led-ul semnalizator

Schema electrică

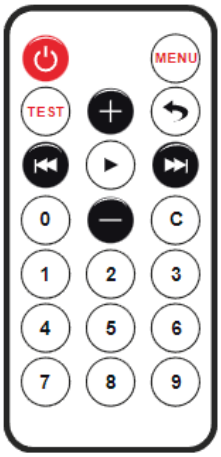


Perifericele utilizate în proiect sunt 2 micro-servomotoare, un ecran LCD de configurație I2C, o telecomandă de 38kHz și un receptor cu infraroșu și un led. Servomotoarele reprezintă ușile mașinii, respectiv geamurile. Led-ul va semnaliza intrarea în stările sistemului, ce vor fi afișate pe ecranul LCD, în momentul apăsării tastelor corespunzătoare de pe telecomandă. Servomotoarele sunt alimentate cu 5V/3.3V fiecare, ecranul LCD cu 5V iar receiver-ul la fel.

3.1 Stările sistemului



4. Descrierea componentelor necesare



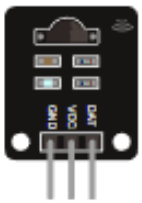
- **Telecomandă cu infraroșu**

O telecomandă cu infraroșu de 38 KHz cu 20 de taste funcționale. Utilizată împreună cu modulul receptor IR.

Tastele trimit mesaje în infraroșu codificate utilizând formatul NEC .Fiecare tastă trimite o valoare de comandă diferită iar câmpul de adresă este întotdeauna 0.

- **Receiver cu infraroșu**

Receptor infraroșu 38KHz folosit împreună cu telecomanda.



Are 3 pini : GND , VCC (Voltaj pozitiv) , DAT (Digital Output).

Receptorul trimite comenzi cu infraroșu.

GND-la Ground Arduino, VCC la 5V de pe Arduino iar DAT la pinul 4 .

Specificații (Telecomanda + Receiver)

-Modul IR pentru recepție cu led,fir și telecomandă ce permite controlul la distanță a diferitelor proiecte în Arduino și alte sisteme

-Tensiune de alimentare : 5VDC

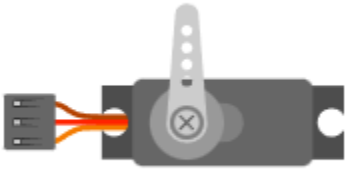
-Alimentare telecomanda: CR2025 inclusă în pachet

-Distanță transmisie de până la 8 metri

-Unghi transmisie de până la 60 grade

-Transmisie de 38KHz

- **Micro-servo motor standard**



Pin names

Name	Description
PWM	Servo control signal
V+	Positive voltage (5V)
GND	Ground

Specificații:

-Servo motor tip SG90, greutate 9grame

-Rotație servomotor între 0 și 180 de grade

-Acest tip de servomotor este fără limitator de cursă, acest lucru însemnând că atunci când nu este alimentat, servomotorul se poate roti de mană în mod continuu

-Tensiune de alimentare de 3-7.2 VDC

-Unghi actionare de 0-180 de grade

-Dimenisuni mm : 22x 11.5 x 22.5

-GND- la Ground Arduino, V+ la 5V/3.3V , iar pinul PWM la pinul 5(pentru uși) , respectiv pinul 3(pentru geamuri), ambele pinuri fiind digitale pentru pwm.

- **Ecran LCD1602**

Un LCD cu 2 linii, 16 caractere pe linie.

LCD1602 vine în 2 configurații posibile: configurație I2C și configurație standard. Configurația I2C este de obicei mai simplu de utilizat.

Următorul tabel rezumă diferențele cheie:

Property	Standard	I2C
Number of Arduino I/O pins	7*	2 (SCL)/SDA
Backlight control	Optional	Yes
Library name	LiquidCrystal	LiquidCrystal_I2C

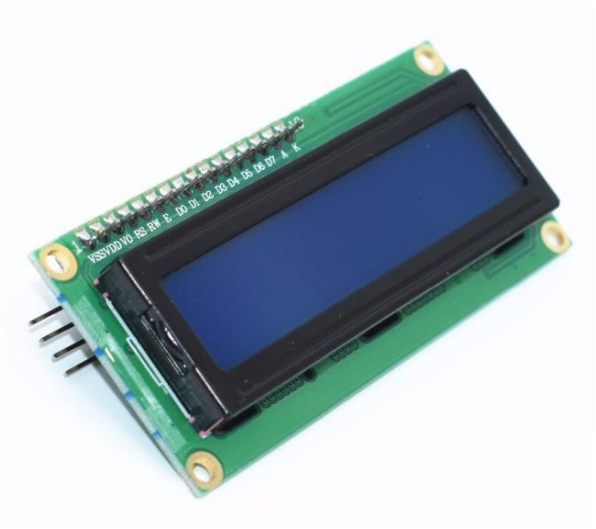
Controlul luminii de fundal necesită un alt pin I/O.

Puteți selecta configurația dorită setând atributul pini. Setați-l la „i2c” pentru configurația I2C sau „full” pentru configurația standard (implicit).

Configuratia pe care o voi folosi pentru acest proiect este I2C.

I2C configuration

Name	Description
GND	Ground
VCC	Supply voltage
SDA	I2C data line
SCL	I2C clock line



Specificații:

-Culoare backlight : Albastru

-Controller: HD44780

-Tensiune alimentare : 5VDC

-**GND la Ground Arduino, VCC la 5 V de pe Arduino, SDA și SCL, la pinii A5,respectiv A4.**

- **Un led de 5mm**

Led standard de 5mm

Name	Description
A	Anode (positive pin)
C	Cathode (negative pin)

Specificatii:

-Diodă LED 5mm

-Culori : Alb, Albastru, Verde, Roșu , Galben

-Tensiune de alimentare : 3-3.4 VDC pentru ledurile Alb,Albastru și Verde iar 2-2.2 VDC pentru ledurile Roșu și Galben

-Unghi vizual de 15-30 grade

-**Anodul conectat la o rezistență ce intră în pinul digital 6, iar catodul în Ground-ul de pe Arduino**

- **O rezistență de 330 Ohm**

5. Implementarea lucrării

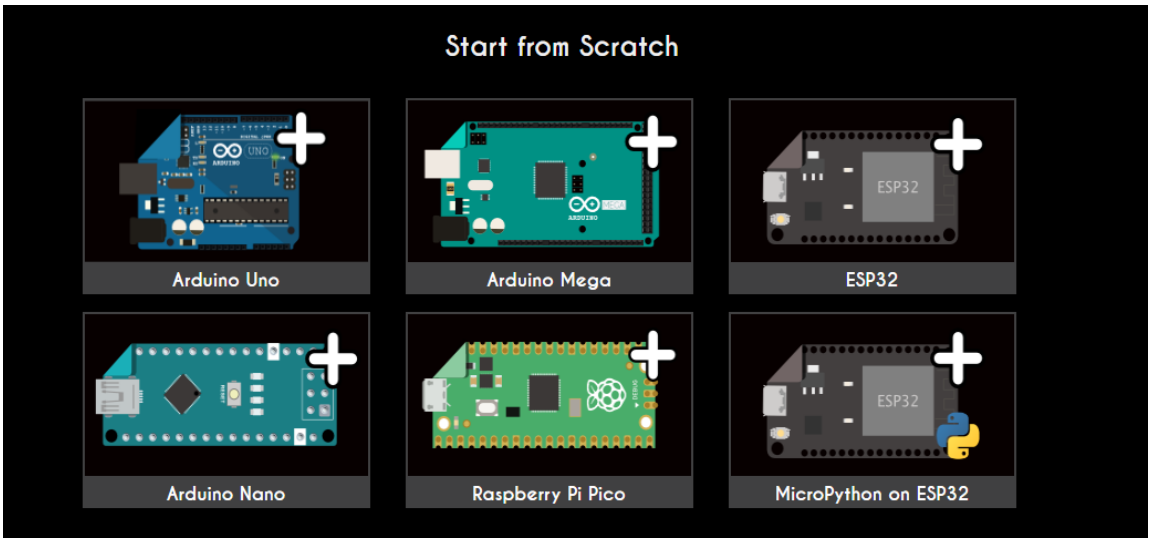
Proiectul prezentat se poate implementa atât fizic cu plăcuța Arduino și piesele descrise mai sus cât și pe un simulator online (precum Wokwi, Tinkercad etc.) .

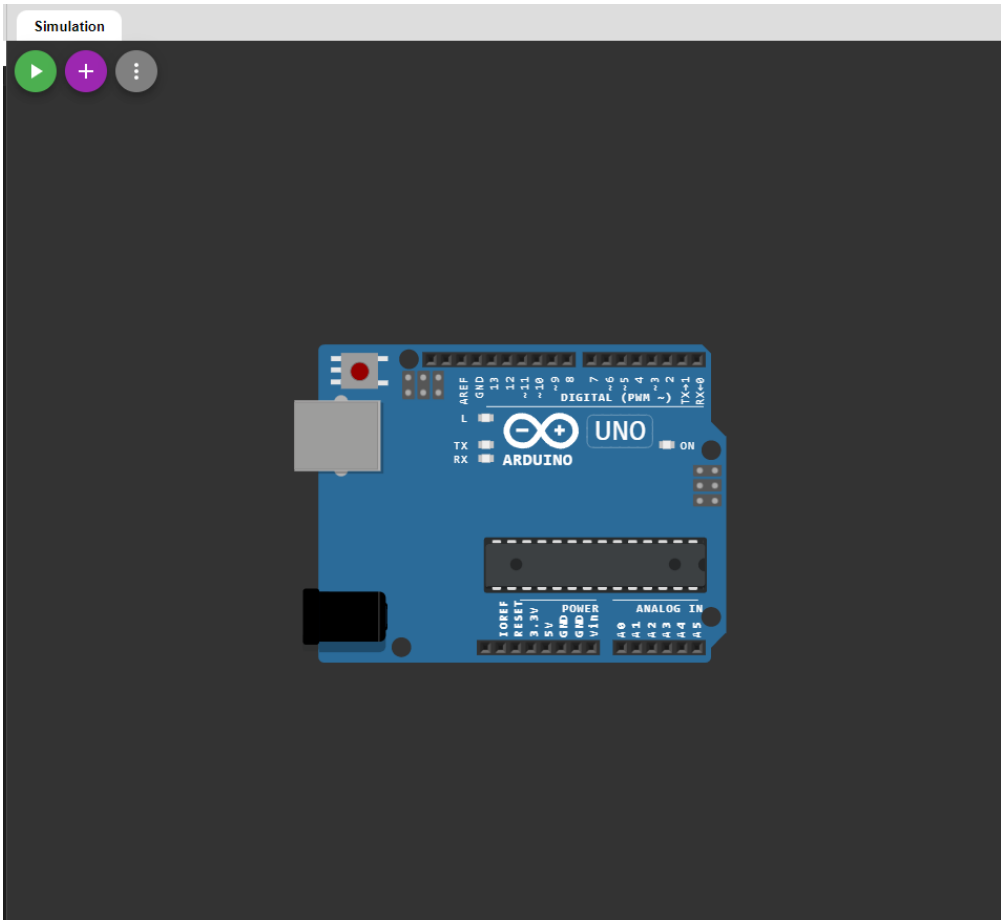
Personal, voi folosi simulatorul online Wokwi pentru simularea funcționalității sistemului dorit. În simulator voi scrie codul, iar in partea dreaptă voi adăuga elementele hardware

specifice (componentele cablate corect la plăcuța Arduino). Ulterior voi compila și rula codul, iar in partea dreaptă a ferestrei voi verifica dacă sistemul se comportă cum ar trebui.

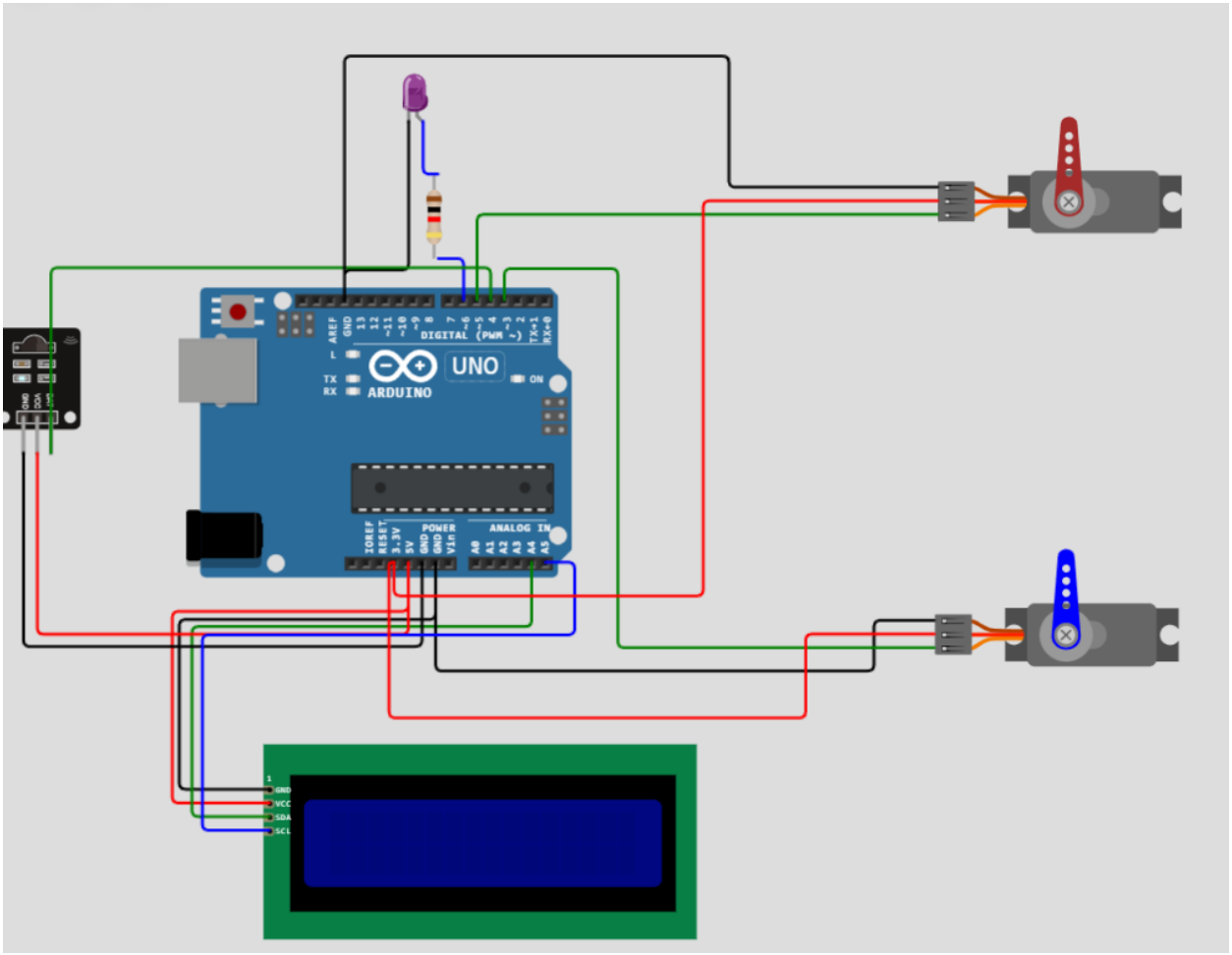
5.1 Secțiuni de cod (Despre implementare)

Intăi de toate voi accesa simulatorul web wokwi, voi alege un proiect nou cu selecția de plăcuța Arduino Uno.





Apoi voi adauga pe rând componentele, apăsând pe “PLUS” sub tab-ul “Simulation”. Si voi realiza cablarea acestora la Arduino.



În final, mă voi apuca de scrierea codului propriu-zis.

1.Voi include librăriile corespunzătoare pentru periferice și anume :

```
#include <LiquidCrystal_I2C.h>
#include <IRremote.h>
#include <Servo.h>
```

2.Declar variabilele necesare și setez parametrii pentru LCD și pinul pentru Receiver-ul IR

```
LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 16 chars
and 2 line display
IRrecv receiver(4);
Servo myservo;  // create servo object to control a servo
Servo myservo1;

/*
"Power" for ON/OFF
"+" for Lock / Double lock
"-" for Unlock
"Pw" = Power
"Dw" = Down ~ -
"Up" = Up ~ +
Brown servo for door
Blue servo for window
1 * (300On phase 700 off phase) for lock
2 * (300On phase 700 off phase) for unlock
3 * (300On phase 700 off phase) for double lock
Door and window 90 degree of spin actuator made in 1 second
*/


int currentState = 0;
int started = 0; // state of the controller
int signal = 0; // signal from the remote controller
float pos = 90; // position of the door actuator
float pos1 = 90; // position of the window actuator
int doorFlag = -1; // flag for locking/unlocking the door
int windowFlag = -1; // flag for locking/unlocking the window
int lightFlagL = -1; // light flag for lock 1 mode
int lightFlagDL = -1; // light flag for lock 2 mode
int lightFlagUn = -1; // light flag for unlock mode
float counterL = 0; // counter for lock 1 mode
float counterDL = 0; // counter for lock 2 mode
float counterUn = 0; // counter for unlock mode

unsigned msMax = 0;

long lightUpEndTime; // TIMERS
long lightUpStartTime; // for lock

long lightUpEndTime1;
long lightUpStartTime1; // for double lock

long lightUpEndTime2;
```

```
long lightUpStartTime2; // for unlock

long doorStart; // for door
long doorEnd;

long windowStart; // for window
long windowEnd;
```

3. Scriu funcțiile pentru LCD (de pornire, de printare a stării)

```
void lcdsetup(){ // lcd start function
  lcd.backlight();
  lcd.setCursor(0,0);
  lcdPrintState();
}

void lcdPrintState(){ // prints the state of the machine => unlocked, locked,
double locked
  lcd.setCursor(0,0);
  if(currentState == 0){
    lcd.print("UNL");
  }
  else if(currentState == 1){
    lcd.print("LCK");
  }
  else if(currentState == 2){
    lcd.print("DLK");
  }
}

void lcdPrint(char* text) // prints text given
{
  lcd.setCursor(0, 1);
  lcd.print(text);
}
```

-pentru Remote/telecomandă (verificarea semnalului transmis de către apăsarea tastei de pe telecomandă corespunzătoare și afișarea stării pe LCD) . Pentru semnalele transmise și codul acestora, m-am folosit de documentațiile oferite de simulatorul WOKWI.

```
void remoteController(int sig){ // use the function above
  if(sig == 162){
  // if(sig == 48){
    lcdPrint("Pw");
  }
  else if(sig == 152){
  // else if(sig == 74){
    lcdPrint("Dw");
  }
  else if(sig == 2){
  // else if(sig == 24){
    lcdPrint("Up");
  }
}
```

- pentru aprinderea ledului pentru câteva secunde,în funcție de stările în care se află sistemul

(300ms pentru on-phase si 700ms pentru off-phase)

```
void lightUpUn(){ // lighting the led for unlock mode
  if(lightFlagUn == 1){

    if(counterUn == 0.19){
      lightUpStartTime2 = millis(); // start timer
    }

    if(0 <= counterUn && counterUn < 300 || 1000 <= counterUn && counterUn <
1300){
      digitalWrite(6, HIGH);
    }
    else if(300 <= counterUn && counterUn < 1000 || 1300 <= counterUn &&
counterUn < 2000){
      digitalWrite(6, LOW);
    }
    else if(counterUn >= 2000){
      lightUpEndTime2 = millis();
      counterUn = 0;
      lightFlagUn = -1;
      // Serial.println(lightUpEndTime2 - lightUpStartTime2);
      // the difference is 2.027 s ~ 2 s (full period)
      }
    counterUn += 0.095;
  }
```

```
}
void lightUpL(){ // lighting the led for lock mode 1
  if(lightFlagL == 1){

    if(counterL == 1){
      lightUpStartTime = millis(); // start timer
    }

    if(0 <= counterL && counterL < 300){
      digitalWrite(6, HIGH);
    }
    else if(300 <= counterL && counterL < 1000){
      digitalWrite(6, LOW);
    }
    else if(counterL >= 1000){
      counterL = 0;
      lightFlagL = -1;
      lightUpEndTime = millis();
      //Serial.println(lightUpEndTime - lightUpStartTime);
      // the difference is 1 second
      }
    counterL += 0.125;
  }
}
```

```
void lightUpDL(){ // lighting the led for lock mode 2
  if(lightFlagDL == 1){

    if(counterDL == 0.30){
      lightUpStartTime1 = millis(); // start timer
    }

    if(0 <= counterDL && counterDL < 300 || 1000 <= counterDL && counterDL < 1300
|| 2000 <= counterDL && counterDL < 2300){
      digitalWrite(6, HIGH);
    }
    else if(300 <= counterDL && counterDL < 1000 || 1300 <= counterDL &&
counterDL < 2000 || 2300 <= counterDL && counterDL < 3000){
      digitalWrite(6, LOW);
    }
    else if(counterDL >= 3000){
      lightUpEndTime1 = millis();
```

```
      counterDL = 0;
      lightFlagDL = -1;
      //Serial.println(lightUpEndTime1 - lightUpStartTime1); // the diff is 3.048
s ~ 3 s period
      }
    counterDL += 0.15;
  }
}
```

- pentru servomotoare (blocarea și deblocarea ușilor și geamurilor, respective celor 2 servomotoare , maro pentru uși și albastru pentru geamuri)

```
-   void doorLocking(){ // locking the door
-   // door lock 1 actuator spin
-
-       if(doorFlag == 1 && pos > 0){ // from 0 mode to 1 => 1,137 s ~ 1 s
-         if(pos == 90){
-         //doorStart = millis();
-         }
-         pos -= 0.01;
-         myservo.write(pos);              // tell servo to go to position in
    variable 'pos'
-
-         if(pos < 1){
-         //doorEnd = millis();
-         //Serial.println(doorEnd - doorStart);
-         pos = 0;
-         myservo.write(0);
-         doorFlag = -1;
-         }
-       }
-
-       if(doorFlag == 0 && pos < 90){ // from mode 1 to mode 0
-         pos += 0.01;
-         myservo.write(pos);    // tell servo to go to position in variable
    'pos'
-         if(pos >= 90){
-           pos = 90;
-           myservo.write(pos);
-           doorFlag = -1;
-         }
-       }
-       else if(doorFlag == 0 && pos > 90){ // from mode 2 to mode 0
-         pos -= 0.01;
-         myservo.write(pos);
```

```
-         if(pos <= 90){
-           pos = 90;
-           myservo.write(pos);
-           doorFlag = -1;
-         }
-       }
-       if(doorFlag == 2 && pos < 180){ // from 1 to 2 => 2 seconds
-         if(-1 <= pos && pos <= 1){
-           doorStart = millis();
-         }
-         pos += 0.0125;
-         myservo.write(pos);
-         if(pos >= 180){
-           doorEnd = millis();
-           //Serial.println(doorEnd - doorStart);
-         }
-       }
-   }
-
-   void windowLocking(){
-     if(windowFlag == 1 && pos1 == 90){
-       windowStart = millis();
-     }
-
-     if(windowFlag == 1 && pos1 > 0){
-       pos1 -= 0.02;
-       myservo1.write(pos1);
-       if(pos1 < 1){
-         windowEnd = millis();
-         //Serial.println(windowEnd-windowStart); // the difference is 1.011
    s ~ 1 s for locking/unlocking the window
-         pos1 = 0;
-         myservo1.write(pos1);
-         windowFlag = -1;
-       }
-     }
-
-     if(windowFlag == 0 && pos1 < 90){
-       pos1 += 0.02;
-       myservo1.write(pos1);
-       if(pos1 >= 90){
-         pos1 = 90;
-         myservo1.write(pos1);
-         windowFlag = -1;
-       } } }
```

4. Funția de setup :

```
void setup() {
  Serial.begin(115200);
  myservo.attach(5); // attaches the servo on pin 5 to the servo object
  myservo1.attach(3);
  receiver.enableIRIn(); // Start the receiver
  lcd.init(); // initialize the lcd
  pinMode(6, OUTPUT);
}
```

- Serial.begin – pentru baud rate
- Setez pinii pentru servomotoare : pinul 5 și 3
- Dau start la receiver si inițializez LCD-ul
- Setez pin-ul de output (LED) la 6

5. Funția de loop:

```
void loop() {
  unsigned int msFromStart = 0;
  unsigned int msFromEnd = 0;
  // Timer for time cycle loop
  msFromStart = millis();




  // Checks received an IR signal
  if (receiver.decode()) {
  signal = receiver.decodedIRData.command;
  receiver.resume();
  }
  if(signal == 162 && started == 0){ // for ON
  // if(signal == 48 && started == 0){ //1 for power
    started = 1;
    lcdsetup();
  }
  else if(signal == 162 && started == 1){ // for OFF
  // else if(signal == 48 && started == 1){
    started = 0;
    lcd.clear();
  }
```

```
if(started == 1){
  if(signal == 2 || signal == 152 || signal == 162){ //plus // minus // power
  // if(signal == 24 || signal == 74 || signal == 48){
    remoteController(signal);
  }
  if(currentState == 0 && signal == 2){
  // if(currentState == 0 && signal == 24){
    doorFlag = 1;
    currentState = 1;
    lcdPrintState();
    lightFlagL = 1;
  }
  else if(currentState == 1 && signal == 2){
  // else if(currentState == 1 && signal == 24){
    windowFlag = 1;
    doorFlag = 2;
    currentState = 2; // state
    lcdPrintState();
    lightFlagDL = 1;
  }
  else if(currentState == 2 && signal == 152){
  // else if(currentState == 2 && signal == 74){
    windowFlag = 0;
    lightFlagUn = 1;
    doorFlag = 0;
    currentState = 0;
    lcdPrintState();
  }
  else if(currentState == 1 && signal == 152){
  // else if(currentState == 1 && signal == 74){
    windowFlag = 0;
    lightFlagUn = 1;
    doorFlag = 0;
    currentState = 0;
    lcdPrintState();
  }
  lightUpUn();
  lightUpL();
  lightUpDL();
  doorLocking();
  windowLocking();

}

//Serial.println(started);
```

```
  //Serial.println(signal);
  //Serial.println(currentState);
  signal = 0;

  msFromEnd = millis();
  msMax = max(msMax, msFromEnd - msFromStart);
  /*
  if(msFromEnd - msFromStart > 1){
    Serial.println(msFromEnd - msFromStart);
  }
  */
  //Serial.println(lightFlagDL);
  //Serial.println(counterDL);
  //Serial.println(msFromEnd - msFromStart);
  //delay(100);
  //Serial.println(pos);}
```

În funcția loop :

- testez și timpul de rulare al programului. Ținta dorită este ca principala rutină a softul-ui să ruleze până în 20ms.
- Se verifică prin codul decodificat trimis de pe telecomandă, în ce funcție se intră și în ce stare ne aflăm (LOCKED, DOUBLE LOCKED sau UNLOCKED)
-

5.2. Performanțe și observații

```
// LCD takes a ton of time displaying characters | minimum 20 sec just for
displaying 3 characters
// maximum loop cycle time is 15ms. BUT on average is 2. 12-13ms for printing on
LCD(changing state's status). 14-15 ms for starting the LCD.
```

-Pentru atingerea unui timp cât mai scurt de rulare , prin teste, am dedus că LCD-ul consumă foarte mult timp pentru afișarea caracterelor. Soluția este afișarea a câtor mai puține caractere pe linie.

-Am testat timpii de rulare pentru fiecare funție în parte (Afișarea caracterelor pe LCD consumă 12-13ms și pornirea LCD-ului consuma 14-15ms). Timpul maxim de rulare al buclei este de 15

ms, dar media este de 2. Testarea timpilor de rulare a funcțiilor am realizat-o cu ajutorul counterelor(pentru uși, geamuri și led). Programul a fost optimizat pentru a rula în timp cât mai scurt, întrucât pentru client este enervant dacă atunci când se apasă o tastă pentru închiderea mașinii ,timpul de așteptare a rezultatului dorit să fie prea mare.

6. Concluzii

Nr curent	Componenta	Cost (cu livrare)	Site de cumpărat	Cost total
1	Ecran LCD 1602 I2C	**27,05 RON**	Cleste.ro	**180,54 RON**
2	Modul IR Receptor + Telecomandă	**22,62 RON**	Emag.ro	
3	2x Servomotor SG90 9g 180 grade	**31,92 RON**	Cleste.ro	
4	Kit Arduino (Breadboard, Leduri, Rezistențe, Arduino, Fire jumperi)	**79,68 RON**	Cleste.ro	
5	20 x Fire Dupont tată-tată 10cm	**19,27 RON**	Emag.ro	

Costurile componentelor sunt reprezentate în tabelul de mai sus.

În concluzie, proiectul a fost realizat cu succes. Dificultățile întâmpinate au fost mai mult pe implementarea fizică decât pe cea din simulatorul online, dar consider că am căpătat experiență mai multă în asamblarea și cablarea componentelor pe plăcuță și în realizarea funcționalității acestora, dar și în programarea cu mediul Arduino. Proiectul de față reprezintă o implementare simplă a unei închideri centralizate, ușor de înțeles și de implementat pentru un începător. Spre deosebire de sistemele complexe actuale la mașini, acest proiect implementează doar funcționalitățile de bază .

7. Bibliografie

Sursa proiectului se poate accesa prin urmatorul link:

https://wokwi.com/projects/337162200759140948

Documentațiile folosite pentru componente utilizate în proiect sunt cele oferite de simulatorul Wokwi.

-pentru telecommandă și receiver: https://docs.wokwi.com/parts/wokwi-ir-remote și https://docs.wokwi.com/parts/wokwi-ir-receiver

-pentru servomotoare : https://docs.wokwi.com/parts/wokwi-servo

-pentru LCD : https://docs.wokwi.com/parts/wokwi-lcd1602

-pentru Arduino: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

Librăriile utilizate de program sunt cele din figura de mai jos.

8. Anexă (Codul sursă și explicații segmente cod importante)

```
#include <LiquidCrystal_I2C.h>
#include <IRremote.h>
#include <Servo.h>

LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 16 chars and 2 line display
IRrecv receiver(4);
Servo myservo;  // create servo object to control a servo
```

```
Servo myservo1;

/*
"Power" for ON/OFF
"+" for Lock / Double lock
"-" for Unlock
"Pw" = Power
"Dw" = Down ~ -
"Up" = Up ~ +
Brown servo for door
Blue servo for window
1 * (300On phase 700 off phase) for lock
2 * (300On phase 700 off phase) for unlock
3 * (300On phase 700 off phase) for double lock
Door and window 90 degree of spin actuator made in 1 second
*/


int currentState = 0;
int started = 0; // state of the controller
int signal = 0; // signal from the remote controller
float pos = 90; // position of the door actuator
float pos1 = 90; // position of the window actuator
int doorFlag = -1; // flag for locking/unlocking the door
int windowFlag = -1; // flag for locking/unlocking the window
int lightFlagL = -1; // light flag for lock 1 mode
int lightFlagDL = -1; // light flag for lock 2 mode
int lightFlagUn = -1; // light flag for unlock mode
float counterL = 0; // counter for lock 1 mode
float counterDL = 0; // counter for lock 2 mode
float counterUn = 0; // counter for unlock mode

unsigned msMax = 0;

long lightUpEndTime; // TIMERS
long lightUpStartTime; // for lock

long lightUpEndTime1;
long lightUpStartTime1; // for double lock

long lightUpEndTime2;
long lightUpStartTime2; // for unlock

long doorStart; // for door
```

```
long doorEnd;

long windowStart; // for window
long windowEnd;


void lcdsetup(){ // lcd start function
  lcd.backlight();
  lcd.setCursor(0,0);
  lcdPrintState();
}

void lcdPrintState(){ // prints the state of the machine => unlocked, locked, double locked
  lcd.setCursor(0,0);
  if(currentState == 0){
    lcd.print("UNL");
  }
  else if(currentState == 1){
    lcd.print("LCK");
  }
  else if(currentState == 2){
    lcd.print("DLK");
  }
}

void lcdPrint(char* text) // prints text given
{
  lcd.setCursor(0, 1);
  lcd.print(text);
}

void remoteController(int sig){ // use the function above
  if(sig == 162){
    lcdPrint("Pw");
  }
  else if(sig == 152){
    lcdPrint("Dw");
  }
  else if(sig == 2){
    lcdPrint("Up");
  }
}
```

```
void lightUpUn(){ // lighting the led for unlock mode
  if(lightFlagUn == 1){

    if(counterUn == 0.19){
      lightUpStartTime2 = millis(); // start timer
    }

    if(0 <= counterUn && counterUn < 300 || 1000 <= counterUn && counterUn < 1300){
      digitalWrite(6, HIGH);
    }
    else if(300 <= counterUn && counterUn < 1000 || 1300 <= counterUn && counterUn <
2000){
      digitalWrite(6, LOW);
    }
    else if(counterUn >= 2000){
      lightUpEndTime2 = millis();
      counterUn = 0;
      lightFlagUn = -1;
      //Serial.println(lightUpEndTime2 - lightUpStartTime2);
      // the difference is 2.027 s ~ 2 s (full period)
      }
    counterUn += 0.095;
  }
}


void lightUpL(){ // lighting the led for lock mode 1
  if(lightFlagL == 1){

    if(counterL == 1){
      lightUpStartTime = millis(); // start timer
    }
```

```
    if(0 <= counterL && counterL < 300){
      digitalWrite(6, HIGH);
    }
    else if(300 <= counterL && counterL < 1000){
      digitalWrite(6, LOW);
    }
    else if(counterL >= 1000){
      counterL = 0;
      lightFlagL = -1;
      lightUpEndTime = millis();
```

```
    //Serial.println(lightUpEndTime - lightUpStartTime);
    // the difference is 1 second
    }
   counterL += 0.125;
  }
}

void lightUpDL(){ // lighting the led for lock mode 2
  if(lightFlagDL == 1){

   if(counterDL == 0.30){
    lightUpStartTime1 = millis(); // start timer
   }

   if(0 <= counterDL && counterDL < 300 || 1000 <= counterDL && counterDL < 1300 || 2000
<= counterDL && counterDL < 2300){
    digitalWrite(6, HIGH);
   }
   else if(300 <= counterDL && counterDL < 1000 || 1300 <= counterDL && counterDL < 2000
|| 2300 <= counterDL && counterDL < 3000){
    digitalWrite(6, LOW);
   }
   else if(counterDL >= 3000){
    lightUpEndTime1 = millis();
    counterDL = 0;
    lightFlagDL = -1;
    //Serial.println(lightUpEndTime1 - lightUpStartTime1); // the diff is 3.048 s ~ 3 s period
    }
   counterDL += 0.15;
  }
}

void doorLocking(){ // locking the door
// door lock 1 actuator spin

   if(doorFlag == 1 && pos > 0){ // from 0 mode to 1 => 1,137 s ~ 1 s
    if(pos == 90){
    //doorStart = millis();
    }
    pos -= 0.01;
    myservo.write(pos);           // tell servo to go to position in variable 'pos'

    if(pos < 1){
```

```
    //doorEnd = millis();
    //Serial.println(doorEnd - doorStart);
    pos = 0;
    myservo.write(0);
    doorFlag = -1;
    }
   }

   if(doorFlag == 0 && pos < 90){ // from mode 1 to mode 0
    pos += 0.01;
    myservo.write(pos);    // tell servo to go to position in variable 'pos'
    if(pos >= 90){
     pos = 90;
     myservo.write(pos);
     doorFlag = -1;
    }
   }
   else if(doorFlag == 0 && pos > 90){ // from mode 2 to mode 0
    pos -= 0.01;
    myservo.write(pos);
    if(pos <= 90){
     pos = 90;
     myservo.write(pos);
     doorFlag = -1;
    }
   }
   if(doorFlag == 2 && pos < 180){ // from 1 to 2 => 2 seconds
    if(-1 <= pos && pos <= 1){
     doorStart = millis();
    }
    pos += 0.0125;
    myservo.write(pos);
    if(pos >= 180){
     doorEnd = millis();    //Serial.println(doorEnd - doorStart);
    }
   }
}

void windowLocking(){
 if(windowFlag == 1 && pos1 == 90){
  windowStart = millis();
 }
```

```
  if(windowFlag == 1 && pos1 > 0){
    pos1 -= 0.02;
    myservo1.write(pos1);
    if(pos1 < 1){
      windowEnd = millis();
     //Serial.println(windowEnd-windowStart); // the difference is 1.011 s ~ 1 s for
locking/unlocking the window
      pos1 = 0;
      myservo1.write(pos1);
      windowFlag = -1;
    }
  }

  if(windowFlag == 0 && pos1 < 90){
    pos1 += 0.02;
    myservo1.write(pos1);
    if(pos1 >= 90){
      pos1 = 90;
      myservo1.write(pos1);
      windowFlag = -1;
    }
  }

}


void setup() {
  Serial.begin(115200);
  myservo.attach(5); // attaches the servo on pin 5 to the servo object
  myservo1.attach(3);
  receiver.enableIRIn(); // Start the receiver
  lcd.init(); // initialize the lcd
  pinMode(6, OUTPUT);
}

void loop() {
   unsigned int msFromStart = 0;
  unsigned int msFromEnd = 0;
  // Timer for time cycle loop
  msFromStart = millis();


  // Checks received an IR signal
  if (receiver.decode()) {
```

```
signal = receiver.decodedIRData.command;
receiver.resume();
}
if(signal == 162 && started == 0){ // for ON
 started = 1;
 lcdsetup();
}
else if(signal == 162 && started == 1){ // for OFF
 started = 0;
 lcd.clear();
}

if(started == 1){
 if(signal == 2 || signal == 152 || signal == 162){
  remoteController(signal);
 }
 if(currentState == 0 && signal == 2){
  doorFlag = 1;
  currentState = 1;
  lcdPrintState();
  lightFlagL = 1;
 }
 else if(currentState == 1 && signal == 2){
  windowFlag = 1;
  doorFlag = 2;
  currentState = 2; // state
  lcdPrintState();
  lightFlagDL = 1;
 }
 else if(currentState == 2 && signal == 152){
  windowFlag = 0;
  lightFlagUn = 1;
  doorFlag = 0;
  currentState = 0;
  lcdPrintState();
 }
 else if(currentState == 1 && signal == 152){
  windowFlag = 0;
  lightFlagUn = 1;
  doorFlag = 0;
  currentState = 0;
  lcdPrintState();
 }
```

```
  lightUpUn();
  lightUpL();
  lightUpDL();
  doorLocking();
  windowLocking();

 }

 //Serial.println(started);
 //Serial.println(signal);
 //Serial.println(currentState);
 signal = 0;

 msFromEnd = millis();
 msMax = max(msMax, msFromEnd - msFromStart);
 /*
 if(msFromEnd - msFromStart > 1){
  Serial.println(msFromEnd - msFromStart);
 }
 */
 //Serial.println(lightFlagDL);
 //Serial.println(counterDL);
 //Serial.println(msFromEnd - msFromStart);
 //delay(100);
 //Serial.println(pos);
}



// plus for lock / double lock
// minus for unlock
// LCD takes a ton of time displaying characters | minimum 20 sec just for displaying 3 characters
// maximul loop cycle time is 15ms. BUT on average is 2. 12-13ms for printing on LCD(changing state's status). 14-15 ms for starting the LCD.
```