**UNIVERSITATEA POLITEHNICA TIMIȘOARA**
**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

# SISTEME ÎNCORPORATE

## Sistem de alarmă bazat pe microcontroller, tastatură, ecran, LCD, buzzer, senzori de prezență de tip PIR și servomotor – “passive infrared sensors”

Karolyi Andrei Cristian
Ivașcu Ionela Constantina
Calculatoare și tehnologia informației
Anul III

An universitar 2022-2023

# 2. Enunțul proiectului

**Implementarea unui system de alarma bazat pe microcontroller, tastatura, ecran, LCD, buzzer, sensori de prezență PIR și servomotor - "passive infrared sensors" (2 studenți)**

Caracteristici:

- Sistemul va conține 2 senzori de tip PIR pentru detecția persoanelor
- Armarea și dezarmarea sistemului alarmă se va face cu ajutorul unui cod special introdus de la tastatură, armarea alarmei va acționa servomotorul, presupunând că în realitate acesta acționează un sistem de închidere a unei uși;
- Odată armat, sistemul va detecta prezența unei persoane neautorizate în două încăperi (un senzor în fiecare încăpere) și va afișa câte un mesaj corespunzător pe un afișaj LCD, ca de exemplu: „Alerta camera 1", „Alerta camera 2", „Alerta ambele camere", „Zone libere",etc.
- Fiecare alertă afișată pe LCD va fi însoțită și de un semnal sonor caracteristic, cu ajutorul unui buzzer; în cazul prezenței celor două alerte simultan, se vor succeda în buclă cele doua semnale sonore diferite, cu repetitivitate de 2 – 3 secunde;
- Dacă sistemul este dezarmat, se va afișa un mesaj corespunzător pe LCD, iar buzzer-ul va fi oprit chiar dacă senzorii PIR detectează ceva.

Acest proiect constă în activarea unui sistem de alarmă care detectează mișcarea dintr-o încăpere prin intermediul a doi senzori de prezență PIR. Sunt folosite doua plăci Arduino (o placa joaca rolul de Master, iar una de Slave) care comunică prin protocol I2C, și transmit pe ecranul LCD un mesaj de alertă în cazul în care este detectată mișcare într-una din încăperi. Folosim un potențiometru pentru ajustarea contrastului LCD-ului, iar pentru emiterea sunetelor este folosit un Buzzer care emite diferite frecvențe sonore în funcție de care senzor PIR declanșează alarma. La început servomotorul este pe poziție de UNLOCK, iar pentru a arma sistemul va trebui introdusă o parola (1234, hardcodată în cod), apoi se va afișa pe ecran "alarm enabled". După ce a fost introdusă parola, servomotorul se comută pe poziția de LOCK (180 grade). Daca au trecut 20 de secunde de când Masterul Arduino a fost alimentat se afișează mesajul "PIR INIT: DONE" altfel sistemul se află într-o stare de așteptare (adică sistemul nu face nimic, nu afișează nimic) până trec cele 20 de

secunde, necesare calibrării senzorilor PIR (date obținute din specificațiile acestora). După calibrare datele transmise de către cei 2 senzori sunt colectate de către master și în caz de detecție este afișat un mesaj pe LCD și este acționat BUZZER-ul.

Alarma va porni de fiecare dată când detectează mișcare și pentru a opri alarma este nevoie de un RESET la Arduino Master, apoi sistemul va intra într-o stare „NOT PROTECTED”.

# 3. Descrierea plăcii de dezvoltare utilizată în proiect

**Caracteristici generale:**

Arduino Uno este o placă de microcontroller bazată pe microcontroller-ul ATmega328. Are 14 pini digitali de intrare / ieșire (din care 6 pot fi folosiți ca ieșiri PWM), 6 intrări analogice, un rezonator ceramic de 16 MHz, o conexiune USB, o mufă de alimentare, un antet ICSP și un buton de resetare.

Placa Arduino Uno diferă de toate plăcile anterioare prin faptul că nu folosește cipul FTDI USB-to-serial driver. În schimb, include Atmega16U2 (Atmega8U2 până la versiunea R2) programat ca USB-la-serie convertor.

Revizia 2 a plăcii Uno are un rezistor care trage linia 8U2 HWB la GND, ceea ce face mai ușor de pus în modul DFU.

Revizia 3 a tabloului are următoarele caracteristici noi:

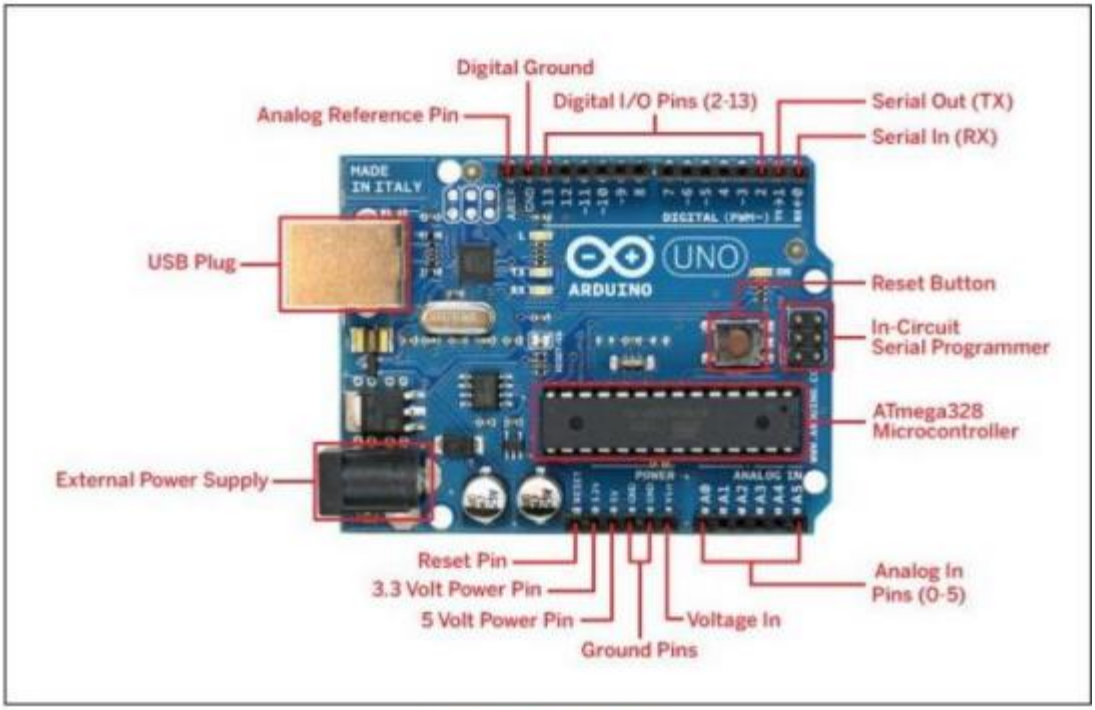
• 1.0 pinout: au fost adăugați pinii SDA și SCL care sunt aproape de pinul AREF și alți doi pini noi plasați aproape de pinul de RESET și de pinul IOREF care permite ecranelor să se adapteze la tensiunea furnizată de placă.

• Circuitul de RESET este mai puternic.

• Atmega 16U2 înlocuiește 8U2.

„Uno” înseamnă 1 în italiană și este numit în acest mod pentru a marca lansarea pe piață a software-ului IDE 1.0 Arduino.

Uno este ultimul dintr-o serie a plăcilor USB Arduino și modelul de referință pentru platforma Arduino.




*Arduino Uno Pin Diagram*

**Arduino Uno R3 particularități:**

• Este un microcontroler bazat pe ATmega328P
• Tensiunea de funcționare a Arduino este de 5V
• Tensiunea de intrare recomandată variază de la 7V la 12V
• Tensiunea i/p (limită) este de 6V la 20V
• Pini digitali de intrare și ieșire: 14
• Pini digitali de intrare și ieșire (PWM): 6
• Pini analogici i/p: 6
• Curentul continuu pentru fiecare pin I/O este de 20 mA
• Curentul continuu utilizat pentru pinul de 3,3V este de 50 mA
• Memorie Flash este de 32 KB, iar 0,5 KB din această memorie este utilizată de boot loader
• SRAM: 2 KB
• EEPROM: 1 KB
• Viteza de CLK este de 16 MHz
• LED încorporat
• Lungimea și lățimea plăcii: 68,6 mm X 53,4 mm
• Greutatea plăcii: 25 g

**Microcontroller Atmel ATmega328P:**

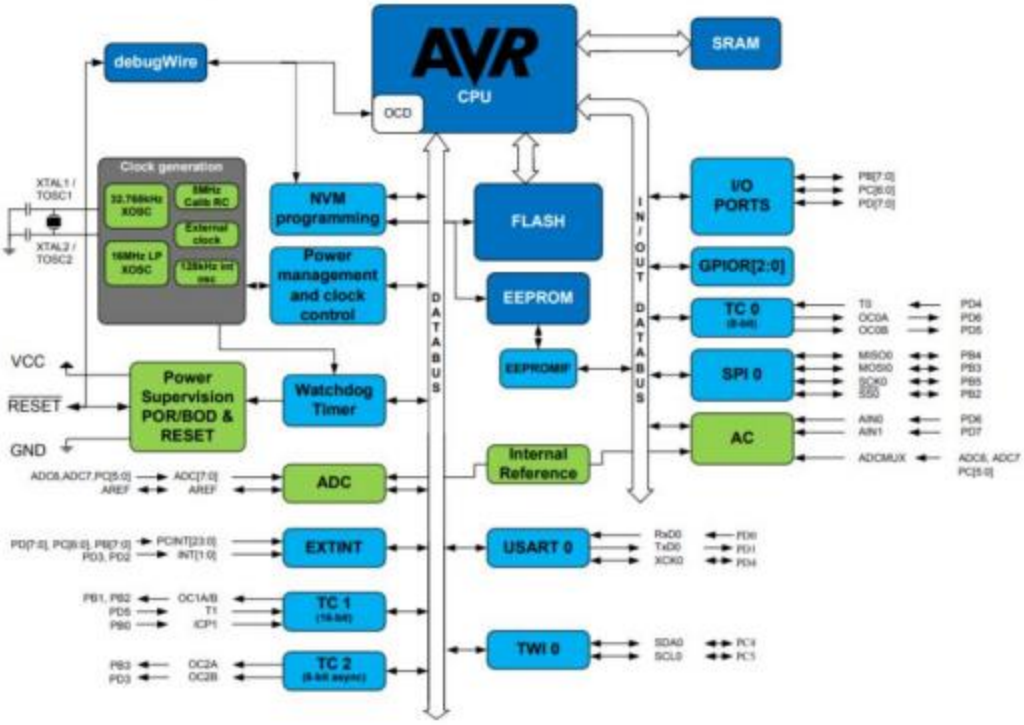
Atmel ATmega328P este un circuit CMOS pe 8 biți cu putere redusă, microcontroller-ul fiind bazat pe arhitectura AVR, care a îmbunătățit arhitectura RISC. Executând instrucțiuni puternice într-un singur ciclu de tact, ATmega328P dă randamente apropiate de 1MIPS/MHz. Aceasta optimizează circuitul pentru consumul de energie în defavoarea vitezei de procesare.

Nucleul Atmel AVR combină un set variat de instrucțiuni cu 32 de registre de uz general. Cele 32 de registre sunt conectate direct la Unitatea Logică Aritmetică (ALU), permițând accesarea a două registre independente într-o singură instrucțiune, care este executată într-un singur ciclu de tact.

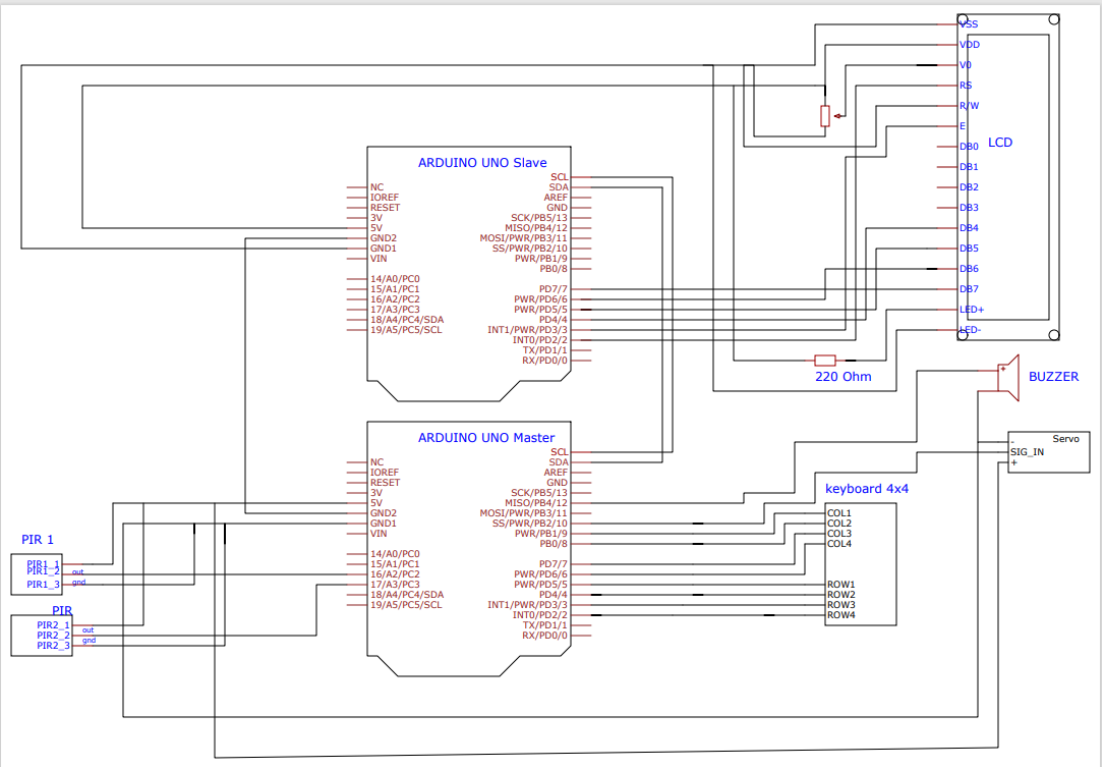
Dispozitivul este fabricat utilizând tehnologia de memorie non-volatilă Atmel cu densitate ridicată. On-chip ISP Flash permite reprogramarea memoriei programului In-System printr-o interfață serială SPI, prin intermediul unui programator convențional de memorie nevolatilă sau de un program de pornire On-chip care rulează pe nucleul AVR

| Features | ATmega328/P |
|---|---|
| Pin Count | 28/32 |
| Flash (Bytes) | 32K |
| SRAM (Bytes) | 2K |
| EEPROM (Bytes) | 1K |
| Interrupt Vector Size (instruction word/vector) | 1/1/2 |
| General Purpose I/O Lines | 23 |
| SPI | 2 |
| TWI ($I^2C$) | 1 |
| USART | 1 |
| ADC | 10-bit 15kSPS |
| ADC Channels | 8 |
| 8-bit Timer/Counters | 2 |
| 16-bit Timer/Counters | 1 |

## Block Diagram



# 4. Arhitectura sistemului




În realizarea proiectului, pentru a lega modulele folosite, am utilizat 2 placi Arduino Uno si 2 breadboard-uri auxiliare. Servomotorul este alimentat la 5V si primeste semnal de la pinul 10 de la Arduino Master. Cei 2 senzori PIR au GND si POWER conectate la GND si 5V de la Arduino Master. Buzzer-ul esti alimentat de la pinul 12 si GND este  conectat la GND de la Arduino Master.
Tastatura este conectata la Placa Arduino Master la pinii 2-9.
Ecranul LCD este conectat la Arduino Slave astfel: VSS la GND, VDD la 5V. V0 este conectat la potentiometru, care are cele 2 capete conectate la GND respective 5 V, RS la pinul 2, R/W la GND, E la pinul 3, iar DB4-DB7 la pinii 4-7, O rezistenta de 220 de Ohmi are un capat conectat la LED+, iar celalalt la 5V, LED- este conectat la GND.

# 5. Descrierea detaliată a modulelor microcontrollerului care au fost implicate în realizarea proiectului

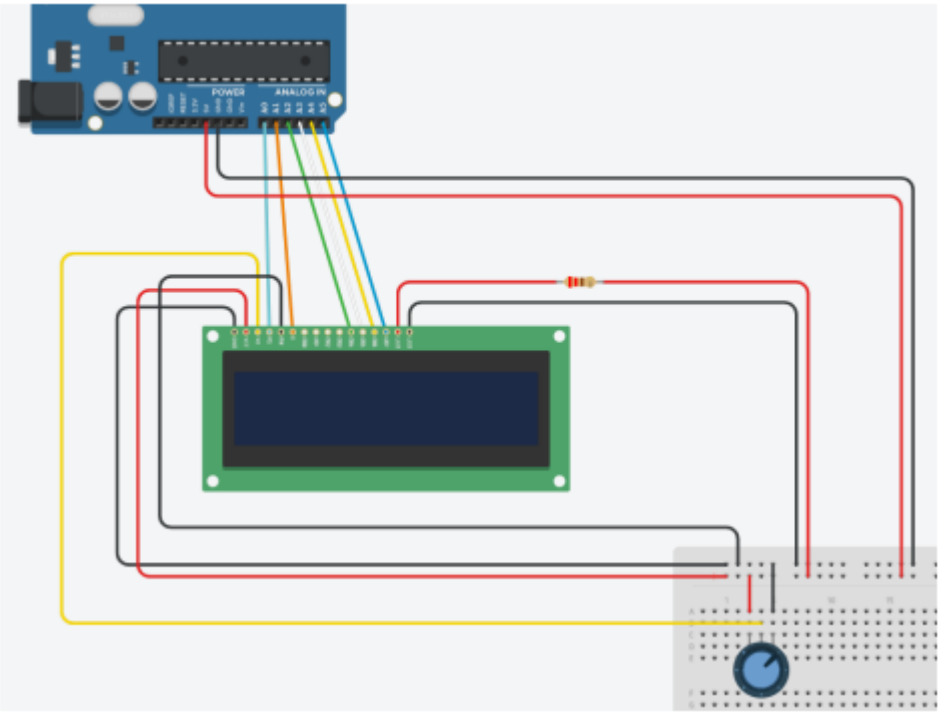
**Modulul I2C:**

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

integrate “master”. Modulul reprezintă un adaptor ce se montează direct pe ecranul LCD. Display-ul trebuie să fie de tip 1602 sau 2004 și bazat pe controller-ul HD44780. Comunicația I2C reprezintă un avantaj, deoarece avem nevoie de doar două fire pentru a comunica cu plăcuța de dezvoltare Arduino sau cu un alt microcontroller. Cele două fire sunt necesare pentru clock și pentru date. Modulul conține și un potențiometru pentru a regla contrastul și este compatibil și cu ecranele ce au iluminare de fundal.

## 6. Funcționalitatea sistemului de afișaj

Sistem de afișaj – LCD 16x2



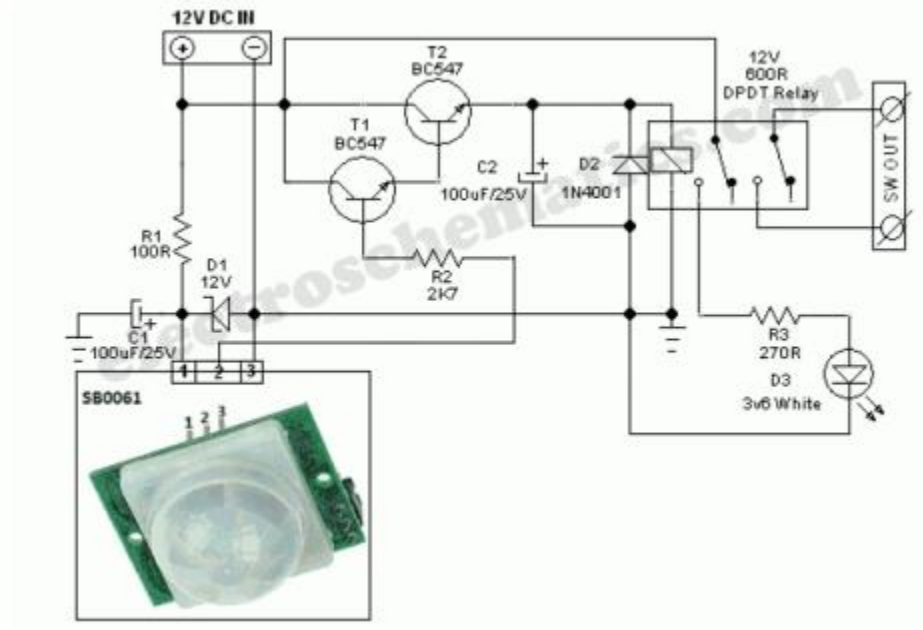
| Număr pin | Simbol | Descriere | Funcție |
|---|---|---|---|
| 1 | Vss | Masă | 0V(GND) |
| 2 | Vcc | Sursă | +5V |
| 3 | V0 | Ajustare contrast LCD | |
| 4 | RS | Selecție registru INSTRUCTION/DATA | Rs=0: INSTR Register<br>Rs=1: DATA Register |
| 5 | R/W | Selecție READ/WRITE | R/W=0: Register WRITE |

| | | | |
|---|---|---|---|
| | | | R/W=1 Register READ |
| 6 | E | Semnal de Enable | Trimitere date către pini când i se aplică puls high-to-low |
| 7 | DB0 | DATA INPUT/ OUTPUT lines | 8BIT: DB0-DB7 |
| 8 | DB1 | | |
| 9 | DB2 | | |
| 10 | DB3 | | |
| 11 | DB4 | | |
| 12 | DB5 | | |
| 13 | DB6 | | |
| 14 | DB7 | | |
| 15 | LED+ | Sursă de tensiune pentru LED+ | +5v |
| 16 | LED- | Sursă de tensiune pentru LED- | 0V |

# 7. Descrierea tehnică a senzorilor și a actuatorilor

1. Senzor PIR (Passive Infrared Sensor)

Schema Block:

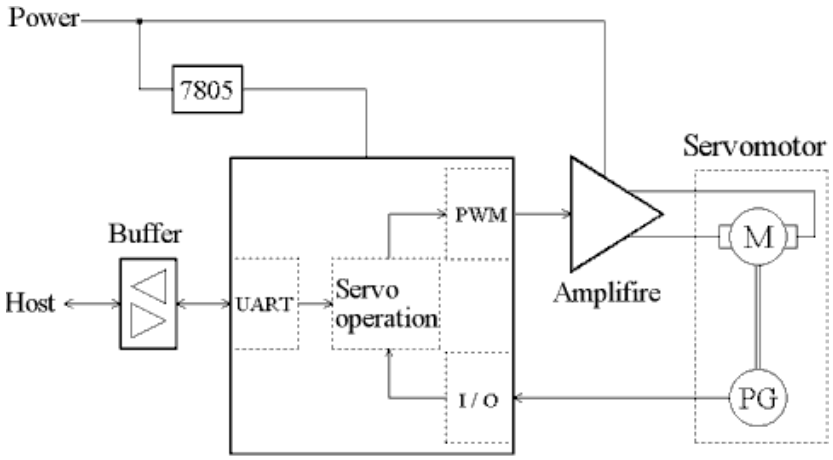



**Principiu de funcționare:**

Atunci când un corp ce emite energie termică într-o formă de radiație infraroșie, de exemplu corpul uman sau animal, va intra în raza senzorului PIR, acesta va detecta o mișcare. De aici vine și numele senzorului pasiv infraroșu (PIR). Acest senzor nu emite nicio energie pentru detectarea ființelor aflate în mișcare, ci doar detectează energia emisă de către acestea, printr-un senzor piezooelectric. Modulul este acoperit de un capac format din lentile Fresnel ce concentrează semnalele infraroșii pe senzorul piezoelectric.

2. Servomotor

Schema bloc:

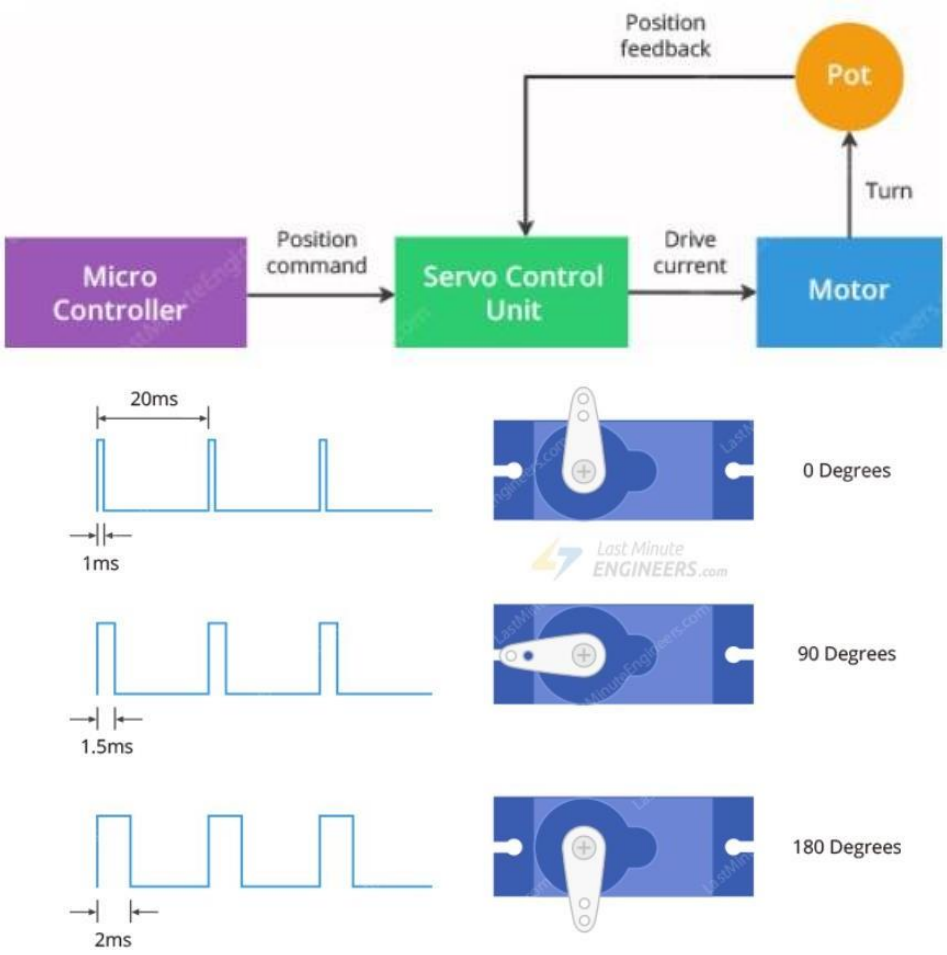



Caracteristici tehnice:

- Tensiune de alimentare: 4.8V - 6V;
- Consum redus de curent;
- Viteza de funcționare: 0.12 s/60o @ 4.8 V;
- Cuplu în blocare la 4.8V: 1.8 kgf*cm;
- Frecvență PWM: 50Hz (conform datasheet-ului anexat);
- Temperatura de funcționare: -30° C - +60° C.

*Principiu de funcționare*

Un servomotor este un dispozitiv electromecanic care produce cuplu și viteză pe baza curentului și a tensiunii furnizate. Un servomotor funcționează ca parte a unui sistem cu buclă închisă, oferind cuplu și viteză, așa cum este comandat de la un microcontroler, utilizând un dispozitiv de feedback pentru a închide bucla. Servo este un termen general pentru un sistem de control cu buclă închisă.




Se poate controla servomotorul trimițând o serie de impulsuri către linia de semnal. Un servomotor analogic convențional se așteaptă să primească un impuls aproximativ la fiecare 20 de milisecunde (adică semnalul ar trebui să fie de 50Hz). Lungimea impulsului determină poziția servomotorului.

# 8. Programul

# MASTER

```
#include <Wire.h>
#include <LiquidCrystal.h>
#include <Servo.h>
#include <Keypad.h>
#include <stdlib.h>
#include <string.h>
```

```
#define SLAVE_PRESENT_CF
#define DEBUG_CF
#define PIR_CALIB_TM_CF 20000U
#define TONE_DURATION_CF 3000
#define FIRST_SENSOR_FREQ_CF 1000
#define SECOND_SENSOR_FREQ_CF 220

#define PASS_LEN 4U +1U
const byte ROWS = 4;
const byte COLS = 4;
const char password[PASS_LEN] = {"1234"};
//----------------pins---------------------
const uint8_t servo_PIN = 10;
const uint8_t PIR_sensor1_PIN = A3;
const uint8_t PIR_sensor2_PIN = A2;
const uint8_t buzzer_PIN = 12;
//-----------------------------------------

enum {
  NOT_PROTECTED,
  PROTECTED,
  ENABLED,
  NOT_ENABLED,
  MUST_INIT
};

struct systemState{
  uint8_t protectionState;
  uint8_t servoState;
```

```
  uint8_t detectionSensorState;
}state;



char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', '4'},
  {'5', '6', '7', '8'},
  {'9', 'A', 'B', 'C'},
  {'D', 'E', 'F', 'G'}
};

byte rowPins[ROWS] = {6, 7, 8, 9};
byte colPins[COLS] = {2, 3, 4, 5};

//index used for comparing entered password
uint8_t index;
char *pass_to_be_checked;
Servo servoModule;

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

#ifdef SLAVE_PRESENT_CF
  void write_status(const char *str)
  {
    Wire.beginTransmission(0x1);
    Wire.write(str);
    Wire.endTransmission();
  }
#else
   void write_status(const char *str)
```

```
  {
    Serial.println(str);
  }
#endif
#ifdef DEBUG_CF
  #define Monitor_Write(x) Serial.println(x)
#else
  #define Monitor_Write(x)
#endif

void setup()
{
  pinMode(servo_PIN, OUTPUT);
  pinMode(PIR_sensor1_PIN, INPUT);
  pinMode(PIR_sensor2_PIN, INPUT);

  Wire.begin();
  write_status(" ");
  servoModule.attach(servo_PIN);
  servoModule.write(0);
  delay(15);
  #ifdef DEBUG_CF
    Serial.begin(9600);
  #endif
  index = 0;

  state.protectionState = NOT_PROTECTED;
  state.detectionSensorState = MUST_INIT;
  state.servoState = NOT_ENABLED;
```

```
  pass_to_be_checked = (char *)malloc(PASS_LEN * (sizeof(char)));
  if(pass_to_be_checked == NULL)
    abort();
}

void loop(){
  if(state.protectionState == NOT_PROTECTED)
  {
    uint8_t index_pass = 0;
    char readKey;
    while(index_pass < PASS_LEN -1)
    {
      readKey = customKeypad.getKey();
      if(readKey)
      {
        Monitor_Write(readKey);
        pass_to_be_checked[index_pass++] = readKey;
      }
    }
    pass_to_be_checked[strlen(pass_to_be_checked)] = '\0';

    if(strncmp(password, pass_to_be_checked, PASS_LEN-1) == 0)
    {
      free(pass_to_be_checked);
      state.protectionState = PROTECTED;
    }
    else
    {
        index_pass = 0;
        write_status("Wrong Password");
```

```
      delay(2000);
      write_status(" ");
    }
  }
  else if(state.protectionState == PROTECTED)
  {
    if(state.servoState == NOT_ENABLED)
    {
      write_status("alarm enabled");
      servoModule.write(179);
      //delay(15);
      state.servoState = ENABLED;
    }
    if(state.detectionSensorState == MUST_INIT)
    {
      //PIR sensors requirement wait for calibration
      if(millis() >= PIR_CALIB_TM_CF)
      {
        state.detectionSensorState = ENABLED;
        write_status("PIR INIT: done");
        delay(2000); //system sleep for lcd clean-up - no risk for using delay here - sensor not read
        write_status(" "); // workaround for lcd to clear screen
      }
    }
    if(state.detectionSensorState == ENABLED)
    {
      uint8_t sensor_value1 = digitalRead(PIR_sensor1_PIN);
      uint8_t sensor_value2 = digitalRead(PIR_sensor2_PIN);

      //Monitor_Write(sensor_value1);
```

```
    Monitor_Write(sensor_value2);

    if(sensor_value1==HIGH && sensor_value2 == HIGH)
    {
      write_status("Alert room 1.Alert room 2");
      for(char i=0; i<15; i++)
      {
        tone(buzzer_PIN, FIRST_SENSOR_FREQ_CF, TONE_DURATION_CF);
        delay(100);
        tone(buzzer_PIN, SECOND_SENSOR_FREQ_CF, TONE_DURATION_CF);
        delay(100);
      }
    }
    else if(sensor_value1 == HIGH)
    {
      write_status("Alert room 1");
      tone(buzzer_PIN, FIRST_SENSOR_FREQ_CF, TONE_DURATION_CF);
    }
    else if(sensor_value2 == HIGH)
    {
      write_status("Alert room 2");
      tone(buzzer_PIN, SECOND_SENSOR_FREQ_CF, TONE_DURATION_CF);
    }
    else {
      write_status(" "); //clear screen
    }
    delay(1000);
   }
  }
}
```

# SLAVE

```
#include <Wire.h>
#include <LiquidCrystal.h>
#include <stdlib.h>

LiquidCrystal lcd(2,3, 4,5,6,7);

short acknowledge_flag;

void onReceive_Handler(int bytes);
void onRequest_Handler(void);

void setup(){
  Wire.begin(0x1);
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 0);
  Wire.onReceive(onReceive_Handler);
  Wire.onRequest(onRequest_Handler);

}
void onReceive_Handler(int bytes)
{

  acknowledge_flag = 0;
  char *str = NULL;
  int index=0;
  char c;

  lcd.clear();

  if(bytes!= 0 && bytes <= 32) //only 32 characters can fill LCD and 1 char = 1 byte
  {
```

```
    str = (char *)malloc((bytes+1) * sizeof(char));
    if(str == NULL) //not enough memory !
      abort();

    while(Wire.available())
    {
      c = Wire.read();
      if( c == '.') //limit of characters per row achived
      {
        str[index] = '\0';
        lcd.write(str);
        lcd.setCursor(0, 1);
        index = 0;
        continue;
      }
      str[index] = c;
      //Serial.println(str[index]);
      index++;
    }
    str[index]='\0';
    lcd.write(str);

    free(str);
    acknowledge_flag = 1;
  }
}
void onRequest_Handler()
{
  Wire.write(acknowledge_flag);
}
void loop(){}
```

# 9. Bibliografie


1. https://store-usa.arduino.cc/products/arduino-zero
2. https://easyeda.com/modules/ATSAMD21G18_72519e2dc4664776b774744553dfdd78
3. https://www.microchip.com/en-us/product/ATsamd21g18
4. https://www.utmel.com/components/atsamd21g18-microcontroller-datasheet-pinout-and-applications?id=438
5. https://arduinogetstarted.com/tutorials/arduino-lcd-i2c
6. http://www.pcserviceselectronics.co.uk/arduino/Ultrasonic/HC-SR04-cct.pdf
7. https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/341-7.pdf
8. Arduino: electronică și programare - 1.Ghid pentru senzor ultrasonic HC-SR04 (google.com)
9. https://www.optimusdigital.ro/ro/motoare-servomotoare/26-micro-servomotor-sg90.html