# Blind Hand Assist □□

**Autor:** Micu Florian-Luis

## Introducere

Acest proiect are ca scop ajutarea oamenilor nevazatori prin aprovizionarea lor cu o metoda de a primi feedback de la mediul inconjurator. Mai exact, un utilizator va purta pe mana device-ul si acesta va putea sa primeasca atentionari in legatura cu proximitatea la care se afla el fata de un obstacol sau poate primii atentionari in legatura cu temperatura obiectelor spre care el isi indreapta mana. Metodele de feedback sunt si ele variate: auditiv sau tactil. Acesti senzori sunt utili pentru o astfel de persoana deoarece lipsa simtului vizual ii depriva de abilitatea de a se orienta in spatiu, drept urmare este important sa constientizeze distanta la care se afla fata de un obiect, iar in cazul in care acestia se pot afla in proximitatea unui obiect incins sau sunt la o distanta sesizabila de un incendiu, vor fi atentionati pentru a se proteja.

## Descriere generală

Se foloseste un intrerupator vertical pentru a comuta intre modul de temperatura (senzorul infrarosu) sau modul de proximitate (senzorul ultrasonic). Totodata, se foloseste un comutator vertical pentru a comuta intre tipurile de feeback: auditiv (piezo buzzer pasiv) sau tactil (motor vibratii). Doua LED-uri sunt folosite pentru a ii ajuta pe cei din jur sa stie ce tip de feedback utilizatorul primeste de la device. De asemenea, senzorul de proximitate are nevoie de date exacte ale temperaturii si umiditatii pentru calculul vitezei sunetului, astfel incat el sa poata prelua distantele corecte indiferent de mediu, din acest motiv device-ul dispune de un astfel de senzor cu acuratete mare (DHT22). Pentru ca acest dispozitiv sa fie portabil, el are atasat o baterie de 9V. De asemenea, device-ul dispune de un buton extern de RESET al placii Arduino NANO, intrucat cel standard este prea mic si greu de apasat.

### Schema Bloc

## Hardware Design

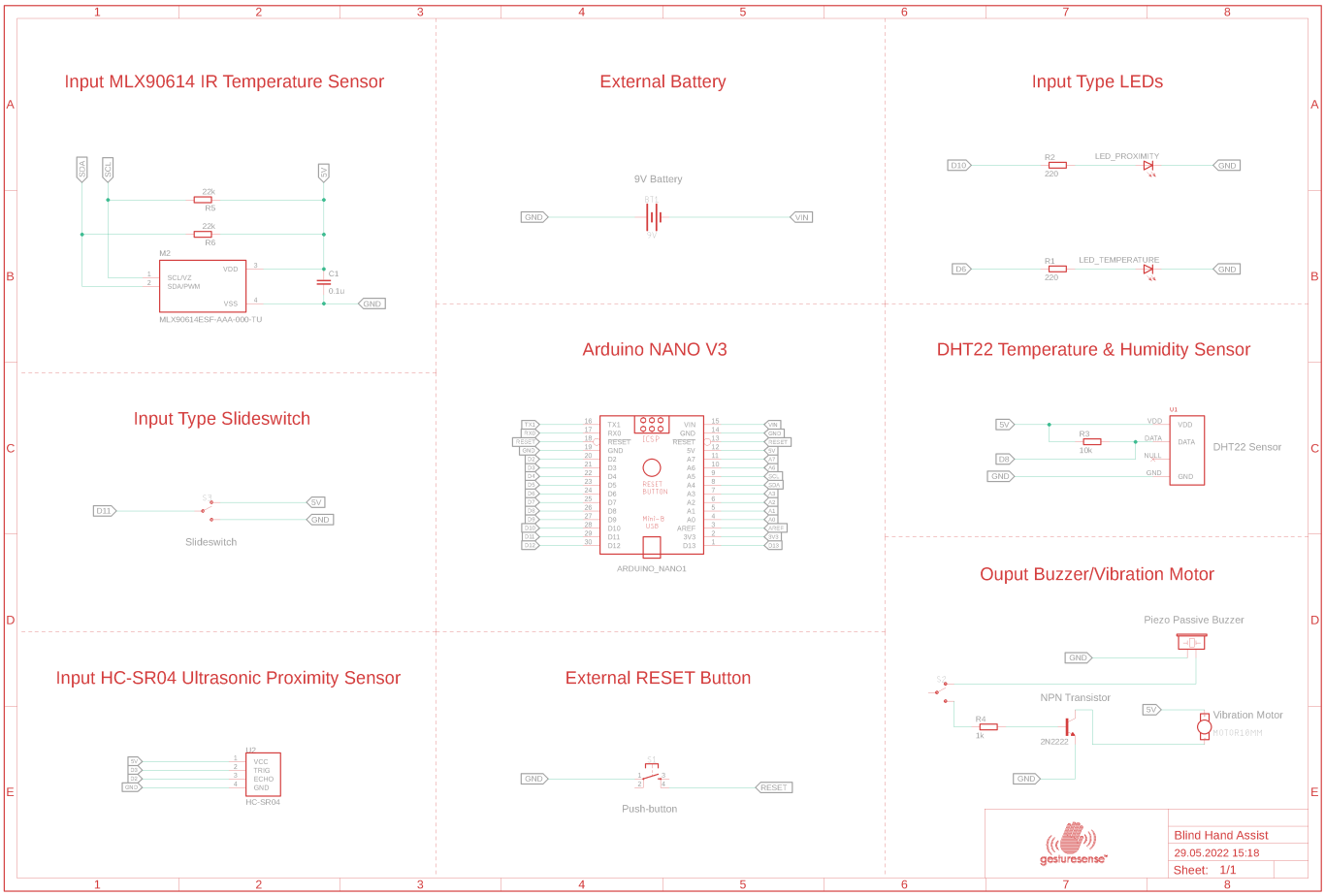
### Lista Componente

1. Arduino Nano V3 (ATmega328P)

2. Senzor proximitate HC-SR04
3. Senzor temperatura si umiditate DHT22
4. Intrerupator vertical x2
5. Tranzistor NPN 2N2222
6. Rezistenta 220Ω x2
7. Rezistenta 1kΩ
8. Rezistenta 10kΩ
9. LED 5mm (verde si albastru)
10. Placa prototipare 70x50mm
11. Piezo Buzzer pasiv
12. Motor vibratii 10mm
13. Senzor Temperatura Infrarosu MLX90614
14. Battery Holder 9V
15. Baterie 9V
16. Cabluri 22AWG
17. Manusa cu arici
18. Barete pini
19. Fire Dupont
20. Buton tip Push-Button

- Preferabil LED-ul sa fie unul cu $V_d$ = 2.2V, altfel rezistenta de 220Ω trebuie schimbata.
- Marimea ventilatorului ar trebui sa fie mica, altfel riscam sa influentam datele senzorului de temperatura si sa gresim calculul proximitatii.
- Grosimea cablurilor a fost aleasa astfel incat sa se poata suda usor pe placa de prototipare.
- Device-ul nu se poate construi in Tinkercad, deoarece nu exista toate piesele necesare.

## Schema Electrica




# Software Design

## Mediul de dezvoltare

Mediul de dezvoltare folosit a fost Android Studio IDE, programarea fiind facuta in C. Biblioteca standard Arduino a fost folosita pentru a usura programarea pe ATmega328P.

## Librarii externe

- DHT by Adafruit
- Adafruit_MLX90614 by Adafruit

## Functii folosite

```
// Receive distance from ultrasonic proximity sensor
void ultrasonic_proximity();

// Sends output signal as a PWM signal to the output source
void send_output_signal(int high, int low);

// Configure Timer1 to stop every 2.8s
void configure_timer1();
```

```
// Enable timer compare interrupt
void init_timer1();

// Reads temperature & humidity every 5.6s using Timer1 intrerupt
ISR(TIMER1_COMPA_vect);

// Configures devices and GPIO pins
void setup();

// Runs continuously the Blind Assist Algorithm
void loop();
```

## Pasi algoritm

1. Se initializeaza Timer1 astfel incat sa se declanseze o intrerupere la fiecare 2.8s:
   1. Se activeaza intreruperile pe Timer1 in functia **init_timer1()**.
   2. Se configureaza registrii lui Timer1 tinand cont de frecventa de operare ATmega328P (16MHz) si frecventa pe care o dorim pentru 2.8s (mai exact 0.35Hz). Aceste date se baga in calculatorul de aici pentru a stabili prescaler-ul si valoarea lui *TOP* in modul CTC. Setarea acestor registri se face in functia **configure_timer1()**.
2. In functia **setup()**, se seteaza pinii de intrare/iesire, se apeleaza functiile pentru activarea si configurarea lui Timer1 si se pregateste comunicarea cu senzorul IR pe interfata I2C.
3. In functia principala, **loop()**, se verifica mai intai pe ce pozitie se afla intrerupatorul vertical pentru a aprinde becul corespunzator tipului de analiza (proximitate sau temperatura):
   1. Daca proximitatea este activata, se apeleaza functia **ultrasonic_proximity()** care calculeaza distanta fata de un obiect prin diferenta dintre timpii de trimitere si primire a unei unde sonore ultrasonice. Deoarece viteza sunetului este relevanta pentru o astfel de unda, la fiecare 2.8s functia asociata intreruperii de pe Timer1 se apeleaza (mai exact **ISR(TIMER1_COMPA_vect)**), iar in aceasta se masoara temperatura si umiditatea prin senzorul DHT22 la fiecare 5.6s pentru a putea calcula viteza sunetului in mediul actual. Daca distanta se afla in parametrii senzorului HC-SR04 (2cm < dist < 400cm) si distanta fata de obiect este mai mica sau egala cu 80cm, atunci se apeleaza functia **send_output_signal(int high, int low)** care trimite un semnal PWM la device-ul de feedback selectat din celalalt switch vertical.
   2. Daca temperatura este activata, se apeleaza functiile standard ale librariei MLX90614 pentru masurarea temperaturii obiectului, iar daca aceasta este mai mare sau egala cu o valoarea de threshold se va trimite un semnal PWM apeland functia **send_output_signal(int high, int low)**.
4. Se asteapta 100ms inainte de urmatorul poll de date pentru a pastra acuratetea

blind_assist_code.c

```
// C++ code
//
#include <DHT.h>
#include <Adafruit_MLX90614.h>

#define echoPin 2
#define trigPin 3
#define DHTPIN 8
```

```
#define DHTTYPE DHT22
#define feedbackPin 5
#define inputTypePin 11
#define proximityLED 10
#define temperatureLED 6
#define MIN_TEMP 27
#define MAX_TEMP 35
#define MAX_DIST 80
#define MIN_DIST 5

// Global variables
DHT dht(DHTPIN, DHTTYPE);
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
float duration, distance, celsius, temp, hum, speed_of_sound = 343;
unsigned char twoSecondsPassed = 2, isProximitySelected = 0;

// Receive distance from ultrasonic proximity sensor
void ultrasonic_proximity()
{
  // Send pulse
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Receive back pulse
  duration = pulseIn(echoPin, HIGH);

  // Measure distance
  distance = (duration / 2) * speed_of_sound / 10000;
}

// Configure Timer1 to stop every 2.8s
void configure_timer1()
{
  // Clear registers
  TCCR1A = 0;
  TCCR1B = 0;
  // Clear internal counter
  TCNT1 = 0;
  // Set top value (CTC mode will compare OCR1A)
  OCR1A = 0xAE63;
  // CTC mode
  TCCR1B |= (1 << WGM12);
  // 1024 prescaler
  TCCR1B |= (1 << CS12) | (1 << CS10);
}

// Enable timer compare interrupt
void init_timer1() {
```

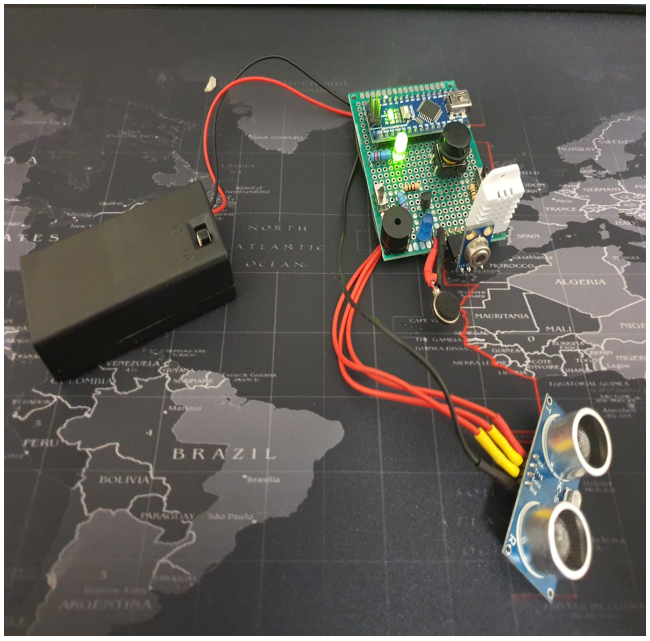
```
  TIMSK1 |= (1 << OCIE1A);
}

// Reads temperature & humidity every 5.6s using Timer1 intrerupt
ISR(TIMER1_COMPA_vect) {
  // If 5 seconds have elapsed -> read temp & humidity
  if (twoSecondsPassed >= 2) {
    temp = dht.readTemperature();
    hum = dht.readHumidity();
    Serial.print("temp: ");
    Serial.println(temp);
    Serial.print("hum: ");
    Serial.println(hum);

    // Recompute speed of sound
    speed_of_sound = 331.4f + (0.606f * temp) + (0.0124f * hum);

    twoSecondsPassed = 0;
  } else {
    ++twoSecondsPassed;
  }
}

// Configures devices and GPIO pins
void setup()
{
  // Disable intreruptions
  cli();

  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(proximityLED, OUTPUT);
  pinMode(temperatureLED, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(feedbackPin, OUTPUT);
  pinMode(inputTypePin, INPUT);
  dht.begin();

  // 2.8s seconds timer initialization
  configure_timer1();
  init_timer1();

  // Enable intreruptions
  sei();

  // I2C
  mlx.begin();
}
```

```
// Sends output signal as a PWM signal to the output source
void send_output_signal(int high, int low) {
  for (int i = 0; i < 100; i++) { // make a sound
      digitalWrite(feedbackPin, HIGH); // send high signal to output
      delay(high); // delay 1ms
      digitalWrite(feedbackPin, LOW); // send low signal to output
      delay(low);
  }
}

// Runs continuously the Blind Assist Algorithm
void loop()
{
  isProximitySelected = digitalRead(inputTypePin);

  // Select which LED to light based on the slideswitch
  if (isProximitySelected) {
    digitalWrite(proximityLED, HIGH);
    digitalWrite(temperatureLED, LOW);
  } else {
    digitalWrite(temperatureLED, HIGH);
    digitalWrite(proximityLED, LOW);
  }

  if (isProximitySelected) {
    // Read distance
    ultrasonic_proximity();

    // Check if distance is valid
    Serial.print("Distance=");
    if (distance >= 400 || distance <= 2) {
      Serial.print("Out of range.\n");
    } else {
      Serial.print(distance);
      Serial.print(" cm.\n");

      // Check if distance is too close to the user
      if (distance <= MAX_DIST) {
        // Create variable frequency based on signal
        int high = map(distance, MIN_DIST, MAX_DIST, 2, 10);
        int low = map(distance, MIN_DIST, MAX_DIST, 1, 3);

        // Send signal to output
        send_output_signal(high, low);
      }
    }
  } else {
    // Read IR temperature
    float tempIR = mlx.readObjectTempC();
```

```
    Serial.print("Ambient = ");
    Serial.print(mlx.readAmbientTempC());
    Serial.print("*C\tObject = ");
    Serial.print(tempIR); Serial.println("*C");

    // Check if temperature is too hot for the user
    if (tempIR >= MIN_TEMP)  {
      // Create variable frequency based on signal
      int high = map(tempIR, MIN_TEMP, MAX_TEMP, 10, 2);
      int low = map(tempIR, MIN_TEMP, MAX_TEMP, 3, 1);

      // Send signal to output
      send_output_signal(high, low);
    }
  }

  // Wait 100 miliseconds before polling data again
  delay(100);
}
```

- Deoarece nu este nevoie sa verificam temperatura si umiditatea des, am dori sa folosim un timer care sa faca intreruperi dupa cel putin 5s. Insa, microcontroller-ul ATmega328P foloseste contorizatori pe 8 biti (Timer0 si Timer2) si un contorizator pe 16 biti, cel din urma dandu-ne voie sa numaram pana la aproximativ 3s. Din aceste motive, am folosit o variabila cu rol de counter care va activa functia de masurare a temperaturii si a umiditati doar atunci cand ne aflam la o intrerupere para.
- Pentru a crea un feedback mai bun, delay-urile HIGH si LOW pentru device-ul de feedback sunt mapate la distanta si respectiv la temperatura, astfel incat frecventa sa creasca atunci cand un obiect se aproprie de utilizator sau cand un obiect devine mai cald. Comportamentul este asemanator unui senzor de parcare auto a carui frecventa creste odata cu aproprierea de un obstacol.

# Rezultate Obţinute




Device-ul sta bine pe mana, nu se incalzeste, feedback-ul motorului de vibratii/buzzer este usor de simtit/auzit, LED-urile se vad clar, iar intrerupatoarele verticale sunt usor de comutat. Butonul de RESET extern e mult mai usor de apasat decat cel standard de pe Arduino NANO, iar bateria externa face device-ul portabil fara sa fie prea greu. Circuitul functioneaza asa cum ar trebui, cand se depasesc valorile de threshold device-ul porneste feedback-ul si toate componentele reactioneaza la schimbarile facute din intrerupatoarele verticale in timp real datorita delay-ului mic.

De mentionat este faptul ca momentan device-ul este prins de manusa cu ajutorul unor benzi cu arici, ceea ce nu este chiar ideal. Pentru o amplasare mai buna s-ar putea printa 3D un holder care sa aiba un locas special pentru o banda cu arici astfel incat device-ul sa fie mai securizat si numarul benzilor cu arici sa scada.

**Demo**: aici

Valorile pentru demo au fost urmatoarele:

- threshold temperatura intre 27.0 si 35.0 grade Celsius
- threshold distanta intre 5.0cm si 80.0cm

# Concluzii

Pe partea de **hardware**, legarea pieselor pe perfboard nu a fost deloc usoara, a fost nevoie de multe iteratii pana sa inteleg cum ar trebui sa creez legaturi mai optime si care sa fie si rezistente, dar la final am evoluat si sunt satisfacut cu sudurile mele. Totodata, am invatat sa citesc mai bine datasheet-urile pieselor ceea ce m-a ajutat enorm la schema electrica cat si la configurarea circuitului. Pentru debug, am folosit un multimetru ca sa masor continuitatea, amperajul si voltajul pe circuit, inainte si dupa fiecare sudura.

Pe partea de **software**, nu am avut deloc probleme deoarece exista tutoriale vaste pentru fiecare

piesa din circuitul meu, iar biblioteciile care le acompaniaza fac toata munca grea (ma refer la senzorul IR care comunica prin protocolul I2C, ceea ce ar fi fost destul de complicat de facut de la zero). Pentru debug, am folosit interfata seriala USART in care mi-am printat anumite valori din program si am incercat si varianta brute-force in care inserez valori si vad fizic cum se comporta device-ul.

Drept urmare, **experienta** de a lucra la acest proiect m-a ajutat foarte mult sa imi dezvolt cunostintele de electronica. Am inteles mai bine cum functioneaza un microcontroller, cum se leaga anumite piese la el si in final am reusit sa imi construiesc propriul device functional, fiind chiar o experienta interesanta.

# Download

Schema Bloc

Schema Electrica

EAGLE biblioteci necesare

Cod Arduino

# Jurnal

- 08.05.2022 → Alegere tema proiect
- 15.05.2022 → Completare Milestone 1: Introducere, Descriere, Schema Bloc si Componente
- 28.05.2022 → Completare Milestone 2: Schema Electrica, Software Design, Rezultate, Concluzii, Bibliografie

# Bibliografie/Resurse

**Resurse Hardware:**

Arduino NANO V3 Pinout

ATmega328P Datasheet

MLX90614 IR Sensor Datasheet

**Resurse Software:**

Arduino Programming Documentation

Export to PDF