Proiect Python

Heart Rate Monitor utilizând ESP32

Studenți:
Asan Aura-Ingrid
Teodorescu Tania-Maria-Gabriela
Grupa 421A

Cuprins

1. Descrierea proiectului

Proiectul are ca scop măsurarea ritmului cardiac cu ajutorul unui senzor de puls și a unui program în MicroPython. Programul va fi încărcat pe un ESP32, iar afișarea bătăilor inimii se va face pe un ecran OLED.

2. Considerente teoretice

- Modelul ESP32

ESP32 este o serie de sisteme cu costuri reduse, cu putere redusă, pe microcontrolere cu cip, cu WiFi integrat și Bluetooth Classic și Low-Energy. Seria ESP32 folosește fie un microprocesor Tensilica Xtensa LX6 în variante dual-core și single-core, microprocesor Xtensa LX7 dual-core sau un microprocesor RISC-V cu un singur nucleu și include comutatoare de antenă incorporate, amplificator de putere, amplificator de recepție cu zgomot redus, filtre și module de gestionare a puterii. ESP32 este creat și dezvoltat de Espressif Systems, o companie chineză cu sediul în Shanghai, și este fabricat de TSMC folosind procesul lor de 40 nm.

Caracteristicile ESP32 includ următoarele:

• Procesoare:
  - CPU: microprocesor Xtensa dual-core (sau single-core) LX6 pe 32 de biți, care funcționează la 160 sau 240 MHz și funcționează la până la 600 DMIPS
  - coprocesor de putere ultra joasă (ULP).

• Memorie: 320 KiB RAM, 448 KiB ROM

• Conectivitate wireless:
  - WiFi: 802.11 b/g/n
  - Bluetooth: v4.2 BR/EDR și BLE (partajează radioul cu Wi-Fi)

• Interfete periferice:
  - 34 pini programabili GPIO (General Purpose Input/Output)
  - 18 canale de conversie analog-digitală (ADC) cu rezoluție de 12 biți
  - 2 canale de conversie digital-analogică (DAC)cu rezoluție pe 8 biți
  - 16 canale de ieșire PWM (Pulse Width Modulation)
  - 10 senzori tactili intern capacitivi
  - 3 interfețe SPI (Serial Peripheral Interface)
  - 3 interfețe UART (Universal Asynchronous Receiver-Transmitter)
  - 2 interfețe I^2C (Inter-Integrated Circuit)
  - 2 interfețe I^2S (Inter-IC Circuit)
  - 1 interfață CAN 2.0 (Controller Area Network)

- controler pentru conectarea dispozitivelor de stocare (carduri de memorie)
- controler gazdă SD / SDIO / CE-ATA / MMC / eMMC
- controler slave SDIO/SPI
- interfață Ethernet MAC cu DMA dedicat și suport planificat IEEE 1588 Precision Time Protocol
- CAN bus 2.0
- telecomandă cu infraroșu (TX/RX, până la 8 canale)
- motor PWM
- LED PWM (până la 16 canale)
- senzor cu efect Hall
- preamplificator analog de putere ultra joasă

• Securitate:

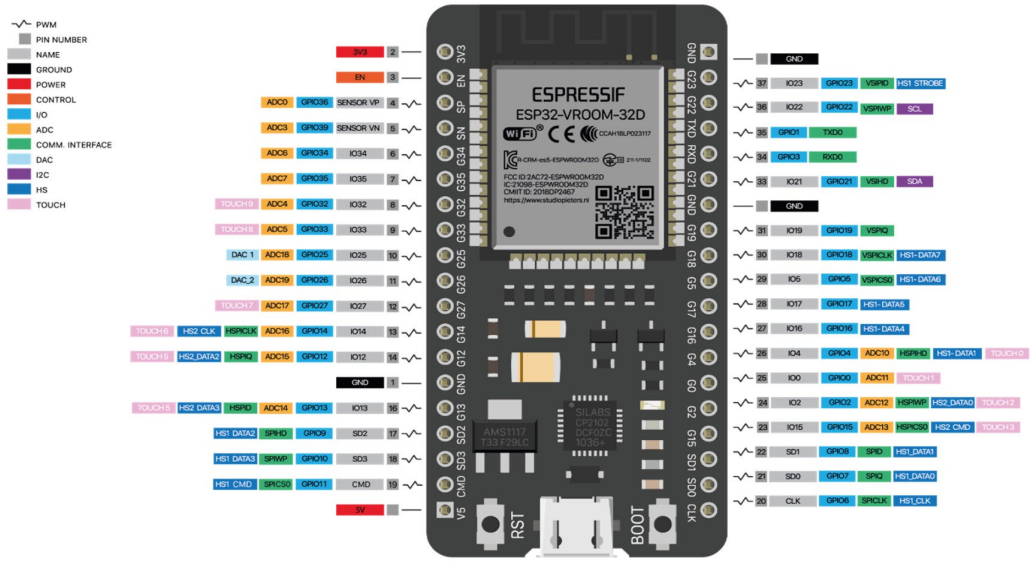
- toate funcțiile de securitate standard IEEE 802.11 sunt acceptate, inclusiv WPA, WPA2, WPA3 (în funcție de versiune) și infrastructura de autentificare și confidențialitate WLAN (WAPI)
- încărcare sigură
- criptare flash
- OTP de 1024 de biți, până la 768 de biți pentru clienți
- accelerare hardware criptografică: AES , SHA-2 , RSA , criptografie cu curbă eliptică (ECC), generator de numere aleatorii (RNG)

• Gestionare a energiei:

- regulator intern cu pierderi reduse
- domeniu de putere individual pentru RTC
- 5 μA curent de somn profund
- trezire de la întrerupere GPIO, temporizator, măsurători ADC, întrerupere senzor tactil capacitiv

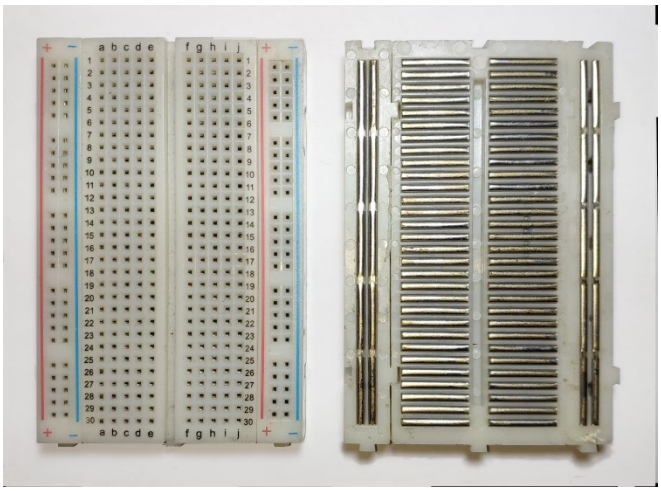
Există mai multe familii de ESP32, cum ar fi: ESP32, ESP32-S2, ESP32-C3, ESP32-S3, ESP32-C6, ESP32-H2 etc.

Modelul ESP32 poate fi integrat în diferite plăci de dezvoltare care, în funcție de producător și tip, pot expune toți pinii/interfețele modulului sau doar o parte din ele. Pentru proiectul nostru am utilizat modulul ESP32-WROOM-32D.



➢ Componente utilizate

Breadboard



Breadboard-ul (placa de prototipare) este folosită pentru realizarea provizorie, rapidă și cu ușurință a circuitelor electrice. Toate componente folosite în acest proiect se conectează pe aceasta. Breadboard-ul constă dintr-o bază din plastic cu orificii mici. Orificiile sunt dispuse în benzi verticale și orizontale. Benzi verticale sunt de obicei conectate electric între ele, în timp ce benzi orizontale nu sunt conectate între ele.

Conexiunile dintre componente le-am realizat cu ajutorul cablurilor Dupont tată-tată, iar pentru a realiza conectarea dintre placa de dezvoltare și computer am folosit un cablu USB tip A - microUSB.

Senzor de ritm cardiac

Senzorul de ritm cardiac, așa cum sugerează și numele, este utilizat pentru a măsura ritmul cardiac. Senzorul funcționează la plasarea degetului pe partea cu inimioară. Când degetul este plasat pe senzor, reflexia luminii LED-ului se schimbă pe baza volumului de sânge din venele capilare, deoarece acesta scade apoi crește la fiecare bătaie a inimii. Această schimbare a luminii LED-ului determină ritmul cardiac.

Caracteristici tehnice:

- Alimentare: 3-5 V
- Consum de curent: 4mA
- Diametru: 16mm
- Compatibil cu Arduino și ESP32
- Ieșire: senzor analogic

Ecran OLED (Organic light-emitting diode)

Pentru afișarea ritmului cardiac am folosit un ecran cu tehnologie OLED 0.96”, iar tipul interfeței este I2C.

Caracteristici tehnice:

- 4 PINI: GND, VCC, SCL, SDA
- Tensiune: 3V ~ 5V DC
- Temperatură de lucru: -30 ℃ ~ 70 ℃
- Rezoluție înaltă: 128 * 32
- Dimensiunile panoului: 26,70 * 19,26 * 1,85mm / 1,03 * 0,76 * 0,07 inch (aprox)
- Zonă activă: 21.74 * 11.2mm /0.86 /0.44 inch (aprox.)
- Driver IC: SSD1306

➢ Limbajul de programare utilizat

MicroPython

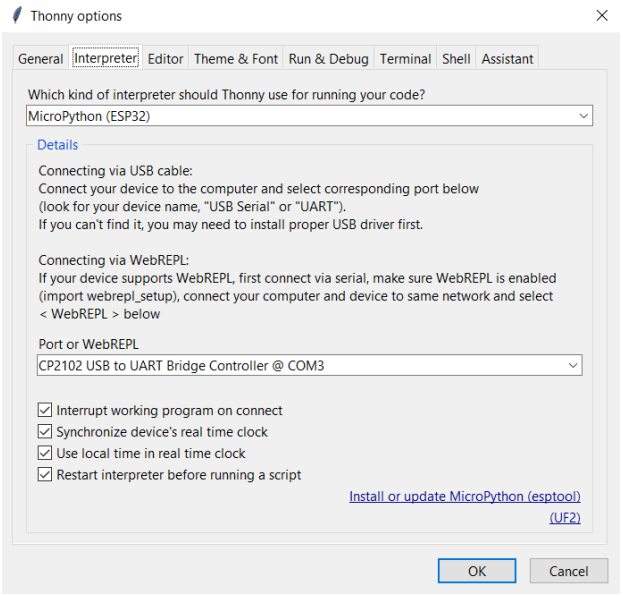
Deoarece în proiect folosim un microcontroler, și anume ESP32, a trebuit să folosim un limbaj de programare dedicat cum este MicroPython. MicroPython este o implementare software a unui limbaj de programare compatibil în mare măsură cu Python 3, care este optimizat pentru a rula pe un microcontroler.

3.1. Recomandări de instalare IDE

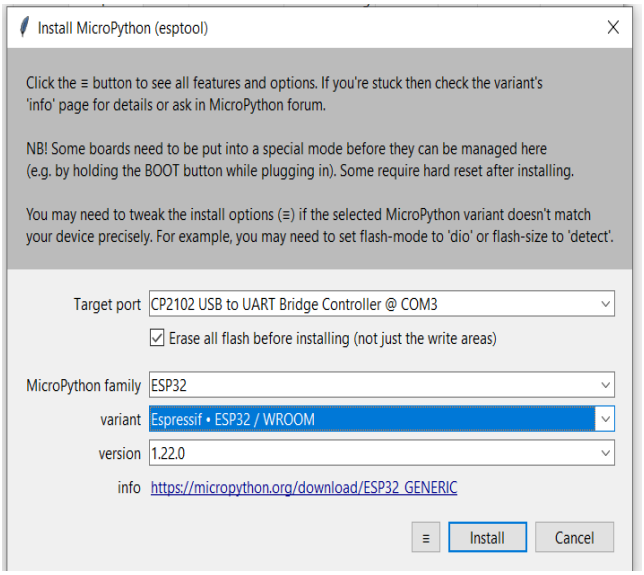
Pentru o bună funcționare a aplicației, recomandăm utilizarea IDE-ului Thonny (Link de instalare: https://thonny.org/).

Se deschide IDE-ul Thonny, iar din bara de sus se accesează meniul Tools -> Manage Plug-ins, se va deschide o nouă fereastră, iar în bara de search se tastează ”esptool” și se instalează.

Tot din bara de sus se mai accesează o data meniul Tools -> Options -> Interpreter și se alege MicroPython (ESP32). Pentru a putea rula MicroPython pe ESP32, din aceeași fereastră (meniul Tools -> Options -> Interpreter) se va apăsa pe ”Install or update MicroPython (esptool)” și se vor completa/alege setăile ca în imaginea 2.



Imaginea 1



Imaginea 2

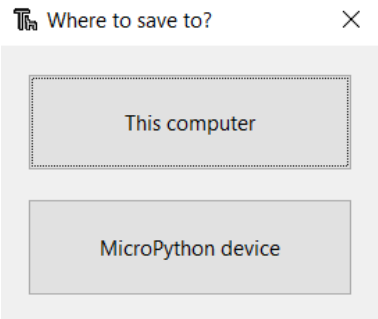
Pentru mai multe detalii legate de instalare și referințe utile, consultați următoarele link-uri:

- https://www.youtube.com/watch?v=elBtWZ_fOZU (instalare Thonny + instalare MicroPython pe ESP32)
- https://www.youtube.com/watch?v=elBtWZ_fOZU&list=PLw0SimokefZ3uWQoRsyf-gKNSs4Td-0k6&pp=iAQB (o mică introducere în MicroPython alături de mai multe aplicații practice interesante)

3.2. Recomandări de instalare driver

Pentru a putea folosi ecranul OLED, este nevoie de driver-ul SSD1306. Cum Thonny IDE nu are acest driver instalat, se va prelua de pe Github astfel:

- se accesează acest repository https://github.com/adafruit/micropython-adafruit-ssd1306/blob/master/ssd1306.py
- se copiază codul pentru ssd1306.py
- se creează un nou File în Thonny în care se va da Paste codului
- se salvează File-ul prin selectarea opțiunii Save As și se selectează MicroPython device cu numele ssd1306.py



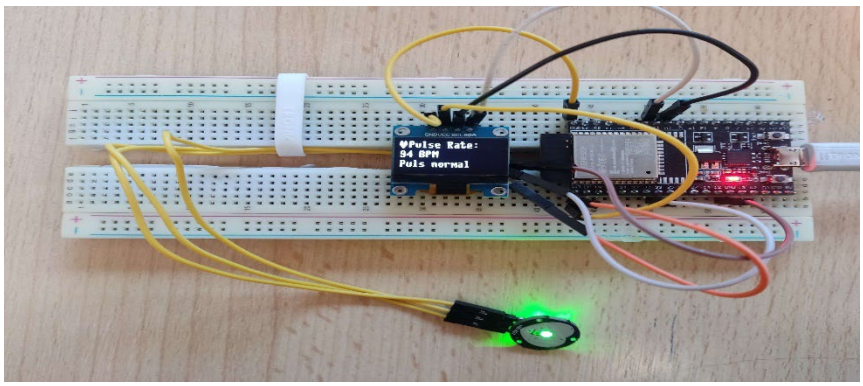
Pentru mai multe detalii, accesați tutorialul https://www.youtube.com/watch?v=Oi_dRctuAQ4

4. Programul și realizarea practică

```
from machine import Pin, ADC, I2C
import ssd1306
import time


# Configurarea pinilor pentru senzorul de puls si ecranul OLED
pulse_sensor_pin = 33
oled_sda_pin = 19
oled_scl_pin = 18


# Initializare ADC pentru senzorul de puls
adc = ADC(Pin(pulse_sensor_pin))
adc.atten(ADC.ATTN_11DB)  # Setam atenuarea la 11dB pentru domeniul maxim de tensiune de 3.3V
```

```
# Initializare ecran OLED
i2c = I2C(0, sda=Pin(oled_sda_pin), scl=Pin(oled_scl_pin), freq=400000)
oled = ssd1306.SSD1306_I2C(128, 32, i2c)

HEART = [
[ 0, 0, 0, 0, 0, 0, 0, 0, 0],
[ 0, 1, 1, 0, 0, 0, 1, 1, 0],
[ 1, 1, 1, 1, 0, 1, 1, 1, 1],
[ 1, 1, 1, 1, 1, 1, 1, 1, 1],
[ 1, 1, 1, 1, 1, 1, 1, 1, 1],
[ 0, 1, 1, 1, 1, 1, 1, 1, 0],
[ 0, 0, 1, 1, 1, 1, 1, 0, 0],
[ 0, 0, 0, 1, 1, 1, 0, 0, 0],
[ 0, 0, 0, 0, 1, 0, 0, 0, 0],
]

def read_pulse():
   # Citire valoare analogica de la senzorul de puls
   pulse_value = adc.read() #o sa citeasca o valoare intre 0-4095

   # Convertire valoare analogica in puls #o persoana poate avea pulsul intre 0-220
   pulse_rate = pulse_value / 18.61 #cum ESP32 citeste o valoare intre 0-4095, 0 va corespunde lui 0,
iar 220 lui 4095
                        #impartim valoarea la 18.61 pentru a afla corespondenta dintre semnal si puls
   return pulse_rate

while True:
   try:  #in caz ca nu merge din cauza unei erori, se va afisa eroarea pe ramura except
      pulse_rate = read_pulse()

      # Golire ecran OLED
      oled.fill(0)
```

```
    # Afisare puls pe ecranul OLED
    oled.text("Pulse Rate:", 11, 1)
    oled.text("{} BPM".format(int(pulse_rate)), 0, 11)
    for y, row in enumerate(HEART):
        for x, c in enumerate(row):
            oled.pixel(x, y, c)
    if(pulse_rate==0):
        oled.text("Mort", 0, 21)
    elif(pulse_rate<50):
        oled.text("Bradicardie", 0, 21)
    elif(pulse_rate<101):
        oled.text("Puls normal", 0, 21)
    elif(pulse_rate<160):
        oled.text("Tahicardie", 0, 21)
    else:
        oled.text("Apelati la 112", 0, 21)

    # Afisare pe ecran
    oled.show()

    # Se asteapta 4 secunde inainte de afisarea urmatoare
    time.sleep(4)

except Exception as e:
    print("Error:", e)
```



Conexiuni hardware

Senzor ~> ESP32:

- S ~> 33
- + ~> 3.3V
- - ~> GND

Ecran OLED ~> ESP32:

- GND ~> GND
- VCC ~> 3.3V
- SCL ~> 18
- SDA ~> 19

5. Bibliografie

- https://en.wikipedia.org/wiki/ESP32#QFN_packaged_chip_and_module
- Documentații PIA
- https://en.wikipedia.org/wiki/Breadboard
- https://cleste.ro/modul-senzor-puls-cardiac.html
- https://cleste.ro/ecra-oled-0-96-inch.html#
- https://en.wikipedia.org/wiki/MicroPython
- https://www.youtube.com/watch?v=Oi_dRctuAQ4
- https://github.com/Gemmus/HeartRateDetector
- https://blog.martinfitzpatrick.com/wemos-heart-rate-sensor-display-micropython/
- https://www.w3schools.com/python/python_try_except.asp
- https://www.w3schools.com/python/python_for_loops.asp
- https://docs.micropython.org/en/latest/library/machine.html
- https://randomnerdtutorials.com/flashing-micropython-firmware-esptool-py-esp32-esp8266/
- https://docs.micropython.org/en/latest/esp32/quickref.html#adc-analog-to-digital-conversion
- https://docs.micropython.org/en/latest/esp32/quickref.html#pins-and-gpio
- https://docs.micropython.org/en/latest/esp8266/tutorial/ssd1306.html
- https://www.engineersgarage.com/micropython-esp8266-esp32-ssd1306/
- https://github.com/adafruit/micropython-adafruit-ssd1306
- https://docs.micropython.org/en/latest/library/machine.I2C.html
- https://randomnerdtutorials.com/esp32-esp8266-analog-readings-micropython/

Link-uri Github:
https://github.com/aura1503/Proiect-PY
https://github.com/TaniaTeodorescu/PROIECT-PYTHON