LABORATORIO DE COMPUTACIÓN I

2do cuatrimestre 2025

TRABAJO PRÁCTICO FINAL

GRUPO 1

Integrantes

Aguirre Christian

Gentta Ivan

Spataro Franco

Aira Nicholas

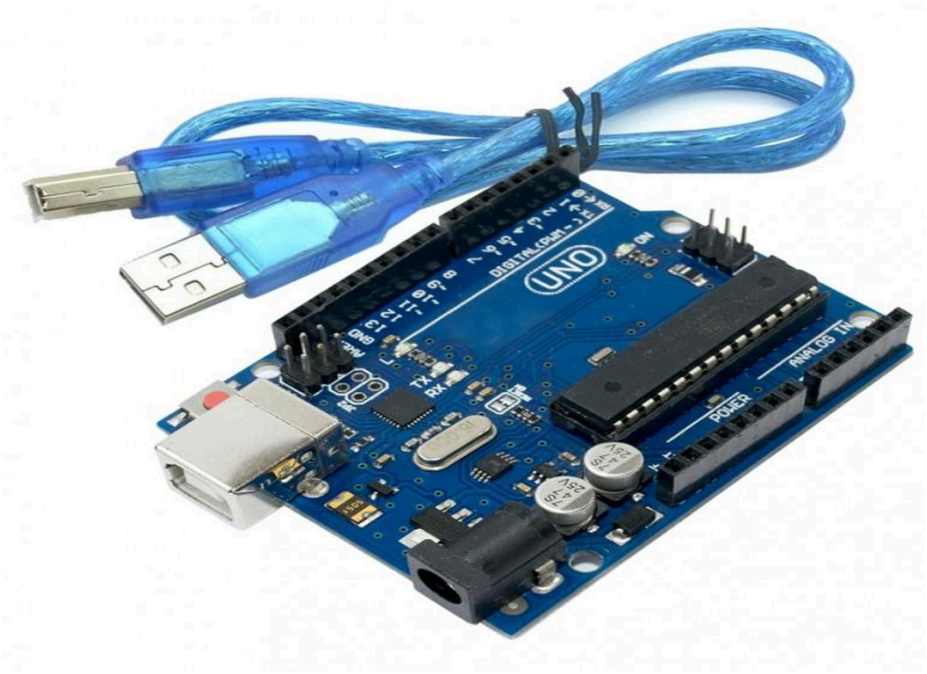
RESUMEN

Este informe presenta detalles técnicos de un dispositivo armado con ARDUINO UNO que toma la temperatura corporal de la mano, los pulsos cardiacos, y los presenta en un display integrado.

ÍNDICE

I. DESCRIPCIÓN TÉCNICA DEL HARDWARE

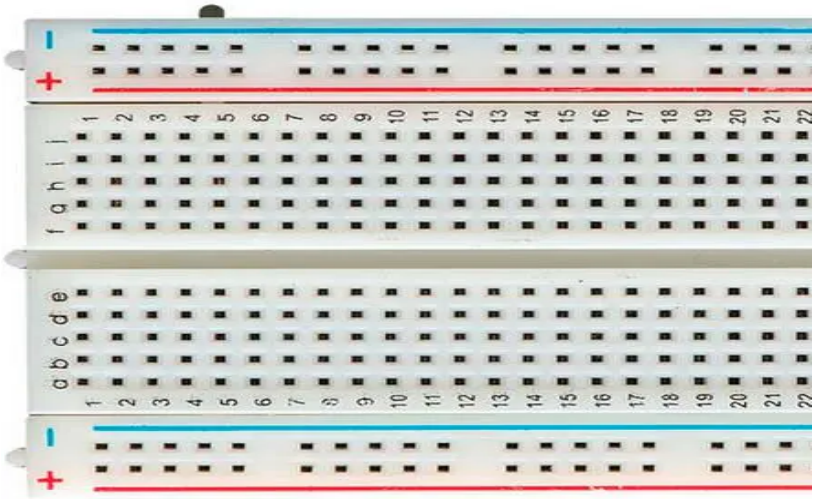
- **Arduino Uno**



El Arduino Uno es una placa de desarrollo basada en el microcontrolador ATmega328P de la familia AVR (fabricado por Microchip). Está diseñada para facilitar el aprendizaje y la creación de proyectos electrónicos y de programación embebida.

Utiliza un voltaje de operación de 5V y posee 14 pines digitales de entrada/salida y 6 analógicos de entrada, para conectar sensores varios, actuadores y módulos. Posee un conector USB tipo B para comunicación y alimentación.

- **Protoboard**

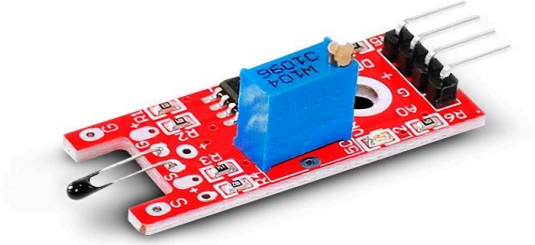


El protoboard es un dispositivo de conexión temporal que permite montar y probar circuitos electrónicos sin necesidad de soldar.

Al insertar un componente o cable en un orificio, éste se conecta eléctricamente con los demás orificios de la misma fila/columna interna.

Su estructura interna cuenta con pistas metálicas interiores que conectan eléctricamente las hileras de agujeros. Posee líneas laterales marcadas con rojo (+) y azul o negro (−) para distribuir voltaje y tierra (GND).

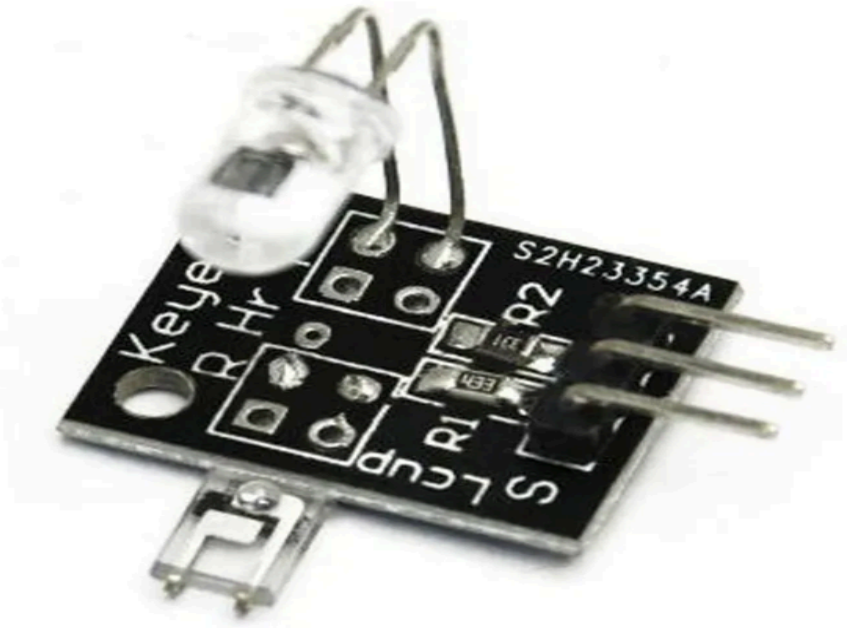
- **Sensor KY-028 de temperatura digital y analógico**



El KY-028 es un módulo sensor de temperatura que combina una salida analógica y una salida digital, lo que permite medir variaciones de temperatura y detectar cambios por encima de un umbral ajustable mediante un potenciómetro.

Salida analógica: Tensión variable proporcional a la temperatura
Salida digital: Señal de nivel lógico (HIGH/LOW) según el umbral configurado
Para su conexión posee 4 pines: *A0* (entrada analógica), *D0* (entrada/salida digital), *GND* (tierra), *VCC* (5V).

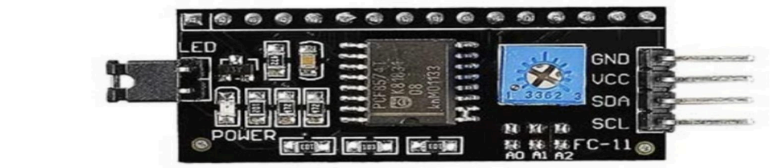
- **Sensor ky-039 de pulso cardíaco.**



El KY-039 es un sensor óptico de pulso cardíaco que utiliza un LED infrarrojo (emisor) y un fototransistor (receptor) para detectar los cambios en el flujo sanguíneo.

Tiene 3 pines de conexión: S (señal, de entrada digital), VCC (5V), GND (tierra).

- **Display LCD i2c**



El LCD I2C es una pantalla de cristal líquido (LCD) que incorpora un adaptador basado en el chip PCF8574, el cual permite comunicarse mediante el protocolo I2C (Inter-Integrated Circuit). Muestra texto en formato matricial de 5×8 puntos por carácter.

Su principal ventaja es que reduce el número de pines necesarios para la conexión, pasando de 16 pines a solo 4 pines (VCC (5V), GND (tierra), SDA (Pin analogico), SCL (Pin analogico).

- **Display OLED 0.91**

El OLED 0.91" es una pantalla de 128×32 píxeles gráfica monocromática basada en tecnología OLED (Organic Light Emitting Diode) y comunicación I2C.

A diferencia de las pantallas LCD, no necesita retroiluminación, ya que cada píxel emite su propia luz, ofreciendo mayor contraste, menor consumo energético y excelente visibilidad incluso en la oscuridad.

Posee los mismos 4 pines que el display LCD: (VCC (5V), GND (tierra), SDA (Pin analogico), SCL (Pin analogico).

- **Keypad 4x4**

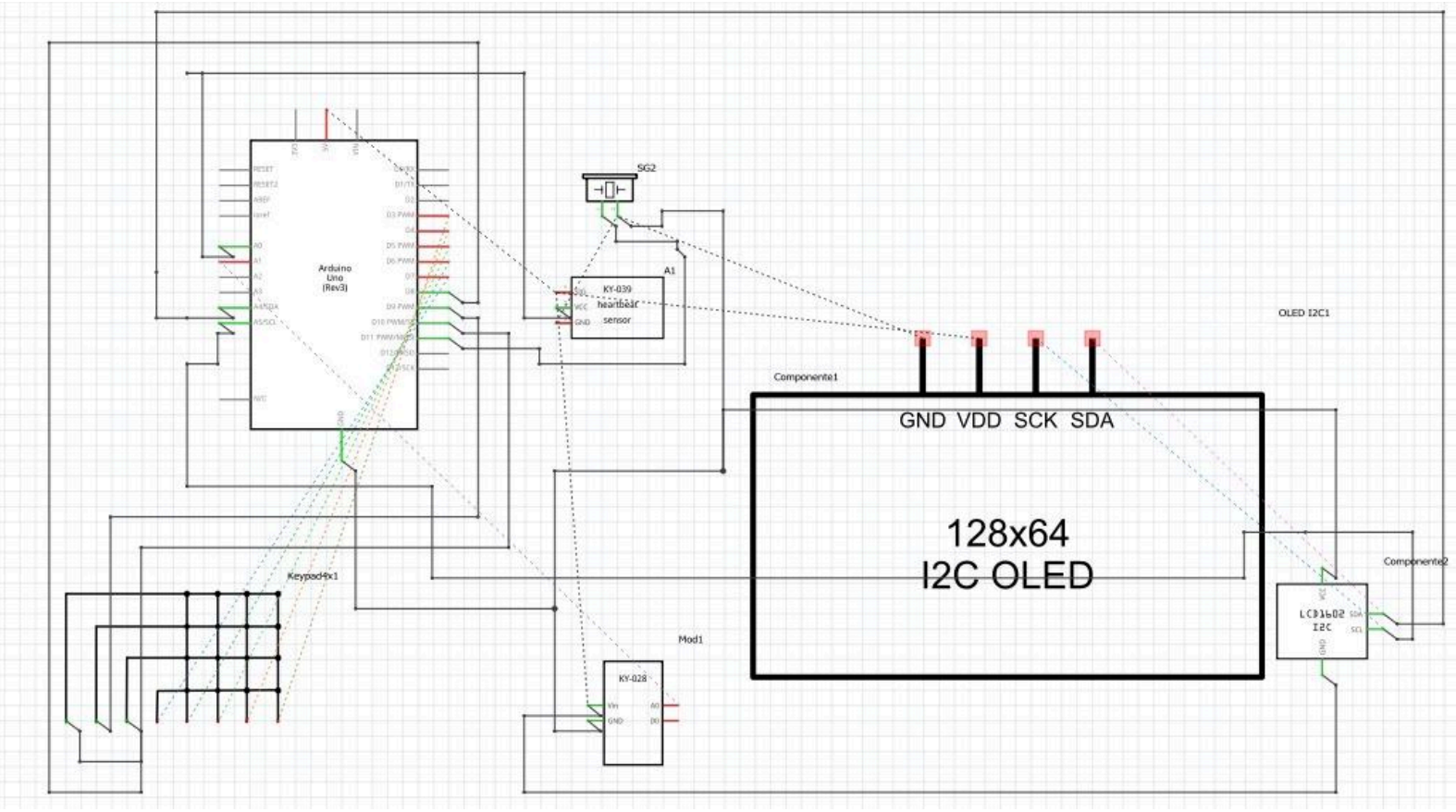
El teclado matricial 4x4 es un dispositivo de entrada que permite capturar pulsaciones de 16 teclas organizadas en una matriz de 4 filas y 4 columnas.
Requiere de 8 pines para su conexión (4 para las filas y 4 para las columnas).

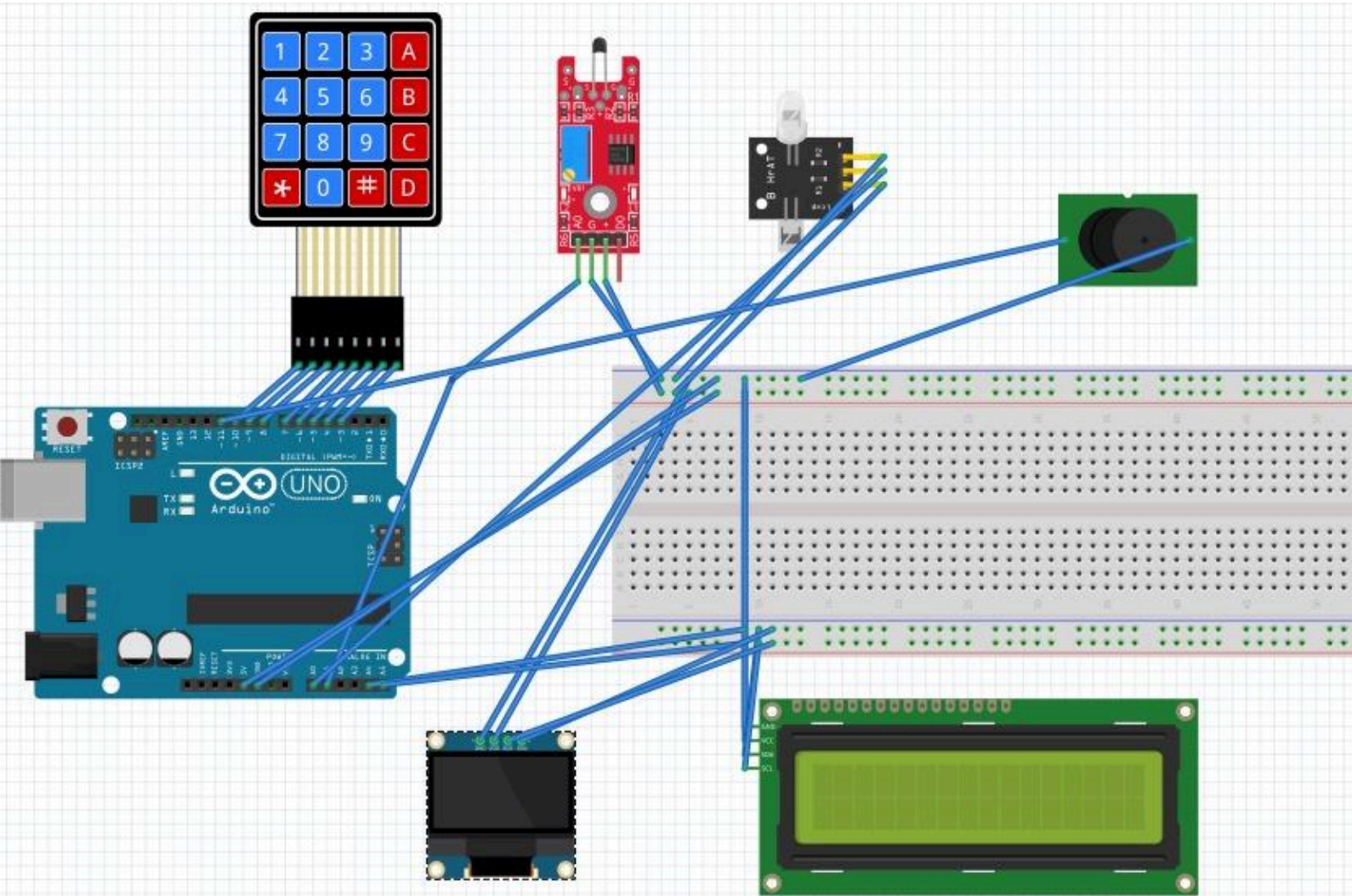
La botonera integra 16 interruptores dispuestos en una malla de 4 filas (R1–R4) y 4 columnas (C1–C4), y sus capas de contacto están hechas de material conductor que se presiona al tocar la tecla. Además posee una cinta flexible para conectar la matriz a los pines de salida.

- **Buzzer**

El buzzer es un dispositivo electroacústico que convierte una señal eléctrica en sonido o vibración audible. Se utiliza en proyectos electrónicos para generar alarmas, tonos, notificaciones o música simple. Sus componentes principales son: Disco piezoeléctrico: vibra al aplicarle corriente alterna, generando el sonido; Caja resonante: amplifica el sonido emitido; Pines de conexión: para alimentación y control desde el microcontrolador mediante un pin digital, y otro para GND (tierra).

II. DIAGRAMA ESQUEMÁTICO O CONEXIONES





III. DESCRIPCIÓN DEL SOFTWARE Y SU ESTRUCTURA

Descripción general

El software fue desarrollado en Arduino IDE (lenguaje C/C++) para controlar un sistema que mide el ritmo cardíaco (BPM) y la temperatura corporal usando sensores analógicos, con visualización de los resultados en una pantalla LCD 16x2 y un display OLED 0.91".

Además, incluye un teclado matricial 4x4 para la selección de funciones y un zumbador (buzzer) que actúa como señal acústica.

El programa está organizado en una máquina de estados, lo que permite manejar distintos modos de funcionamiento sin interrupciones, garantizando lecturas estables y respuestas inmediatas.

1. Estructura general del código

- ***Declaración de variables y configuración inicial (setup):***

  En esta parte se inicializan:

  - ➔ Las entradas y salidas (sensores, buzzer, teclado, pantallas).
  - ➔ Las bibliotecas necesarias (Wire.h, LiquidCrystal_I2C.h, Keypad.h, Adafruit_GFX.h, Adafruit_SSD1306.h).
  - ➔ Los objetos para controlar el display LCD, el OLED y el teclado.
  - ➔ Variables globales que almacenan lecturas, tiempos, y estados del sistema.

  Ejemplo:

```
void setup() {
 lcd.init();
 lcd.backlight();
 display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
 pinMode(buzzer, OUTPUT);
 mostrarMenu();
}
```

- ***Máquina de estados (enum Estado):***

  El código usa una enumeración para definir los modos de trabajo:

  ```
  enum Estado { MENU, SENSOR, TEMP };
  Estado estadoActual = MENU;
  ```

  Esto permite que el sistema sepa qué debe hacer en cada momento:

  > MENU: espera que el usuario seleccione una opción.
  > SENSOR: mide y calcula el BPM.
  > TEMP: realiza la lectura de temperatura.

- ***Bucle principal (loop):***

  Este bloque ejecuta continuamente las siguientes tareas:

  Lectura del teclado:

  Detecta qué tecla se presiona y llama a la función correspondiente ***(manejarMenu())***.

  Control por estado*:*

  Dependiendo del valor de ***estadoActual***, ejecuta:

  medirBPM() *si está en modo SENSOR.*
  probarTemperatura() *si está en modo TEMP.*
  mostrarMenu() *si está en modo MENU.*

  Control por tiempo:

  Se utilizan variables como ***millis()***, ***lastPrint*** e ***interval*** para definir cada cuánto se actualiza el valor mostrado sin detener el resto del programa.

- ***Sensor de pulso (BPM)***

  > Lectura analógica: el sensor KY-039 entrega una señal variable según el paso de la sangre.

  > Detección de flancos: el código compara la lectura actual con la anterior para detectar subidas (flancos ascendentes), lo que indica un nuevo latido.

  > Cálculo de BPM: se mide el tiempo entre flancos (millis() - last_beat) y se calcula el promedio de tres mediciones para suavizar el resultado:

  ```
  print_value = 60000. / (0.4 * first + 0.3 * second + 0.3 * third);
  ```

  > Visualización: cada cierto intervalo, el valor del BPM se muestra en el LCD y se genera un sonido corto con el buzzer.

- ***Sensor de temperatura (KY-028)***

  Cuando el usuario selecciona la opción de temperatura desde el menú, el programa realiza lo siguiente:

  1) El Arduino toma 10 lecturas analógicas consecutivas del sensor.
  2) Calcula el promedio para filtrar el ruido:
  ```
  promedio = suma / 10.0;
  ```
  3) Convierte ese valor analógico a grados Celsius mediante la función ***mapFloat():***
  ```
  temperatura = mapFloat(promedio, 119, 114, 36, 37);
  ```
  4) Muestra el resultado en el LCD durante unos segundos y vuelve al menú principal.

- ***Funciones adicionales:***

  - ➢ **beepTresSegundos()** → activa el buzzer durante 3 segundos.
  - ➢ **mostrarLineaOLED()** → dibuja una línea horizontal en el display OLED (solo decorativo, muestra que el sistema está activo).
  - ➢ **apagarSensor()** → detiene el funcionamiento actual y retorna al menú.

2. Funcionamiento general

1) *El sistema arranca mostrando el menú en la pantalla LCD.*

2) *El usuario elige la función deseada desde el teclado 4x4.*

a) *Si selecciona **BPM**, el sensor mide los latidos y muestra el promedio cada 3 segundos.*

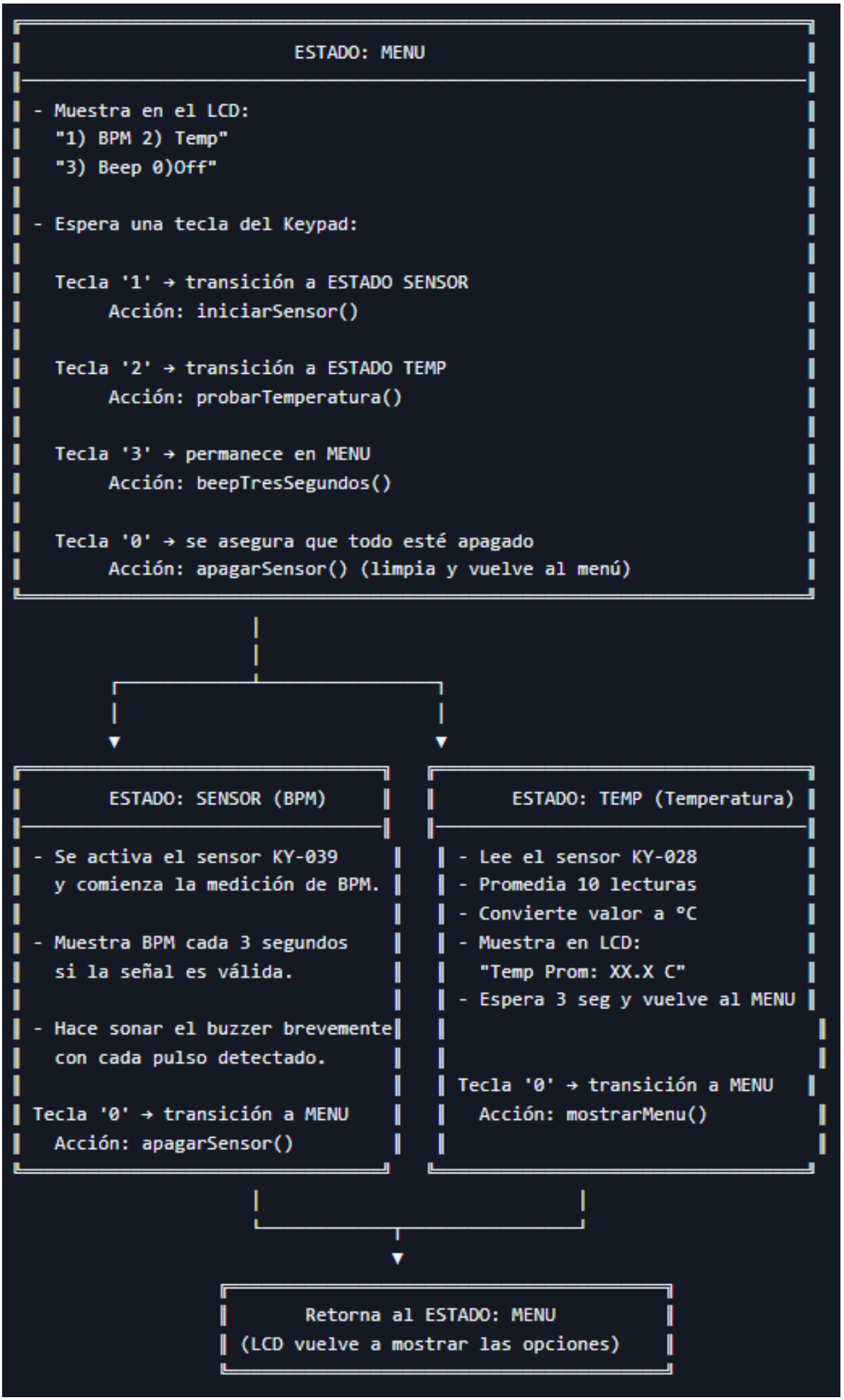
b) *Si selecciona **TEMP**, el sistema mide la temperatura corporal y la muestra en pantalla.*

3) *En cualquier momento puede presionar **"0"** para volver al menú o detener el proceso.*

3. Ventajas del software

- ➢ Permite controlar múltiples sensores desde un solo programa.
- ➢ Usa una máquina de estados eficiente, sin bloqueos.
- ➢ Los tiempos de muestreo y actualización se manejan con millis(), sin detener el sistema.
- ➢ Posee interfaz amigable con menú, buzzer y pantallas.
- ➢ Es modular y ampliable, se pueden agregar nuevas funciones fácilmente.

IV. DIAGRAMA DE MÁQUINA DE ESTADOS



V. DESCRIPCIÓN DEL CONTADOR DE FLANCOS Y CONTROL POR TIEMPO

1. Contador de flancos

Descripción general:

En el proyecto se utiliza un contador de flancos ascendentes para detectar los picos de la señal proveniente del sensor de ritmo cardíaco.

El sistema no cuenta el valor constante de la señal, sino los momentos en que cambia de 0 a 1 (flanco ascendente), que representan cada latido detectado.

Funcionamiento en el código:

En el programa se comparan los valores actuales y anteriores del sensor (last y before).

Cuando el valor actual (last) supera al valor anterior (before), se interpreta como un inicio de subida en la señal, es decir, un flanco ascendente.

```
if (last > before) {
  rise_count++ ;
  if (!rising && rise_count > rise_threshold) {
    rising = true;
    first = millis() - last_beat;
    last_beat = millis();
    print_value = 60000. / (0.4 * first + 0.3 * second + 0.3 * third);
  }
}
```

> rise_count: *cuenta los incrementos de la señal.*

> rising*: indica si el pulso ya fue detectado.*

> rise_threshold: *evita falsos pulsos por ruido.*

(Cada flanco detectado representa un latido cardíaco.)

Función dentro del sistema:

Permite medir con precisión cuántos pulsos reales ocurren, ignorando fluctuaciones pequeñas o ruidos eléctricos del sensor.

2. Control por tiempo (millis)

Descripción general:

El control por tiempo se utiliza para que ciertas acciones se realicen cada cierto intervalo, sin depender del usuario ni del sensor.

En este caso, el tiempo se mide con la función millis(), que devuelve los milisegundos transcurridos desde que se encendió el Arduino.

Funcionamiento en el código:

```
if (millis() - lastPrint >= interval) {
  lcd.clear();
  lcd.print("BPM: ");
  lcd.print(lastBPM, 0);
  lastPrint = millis();
}
```

> millis(): mide el tiempo actual.

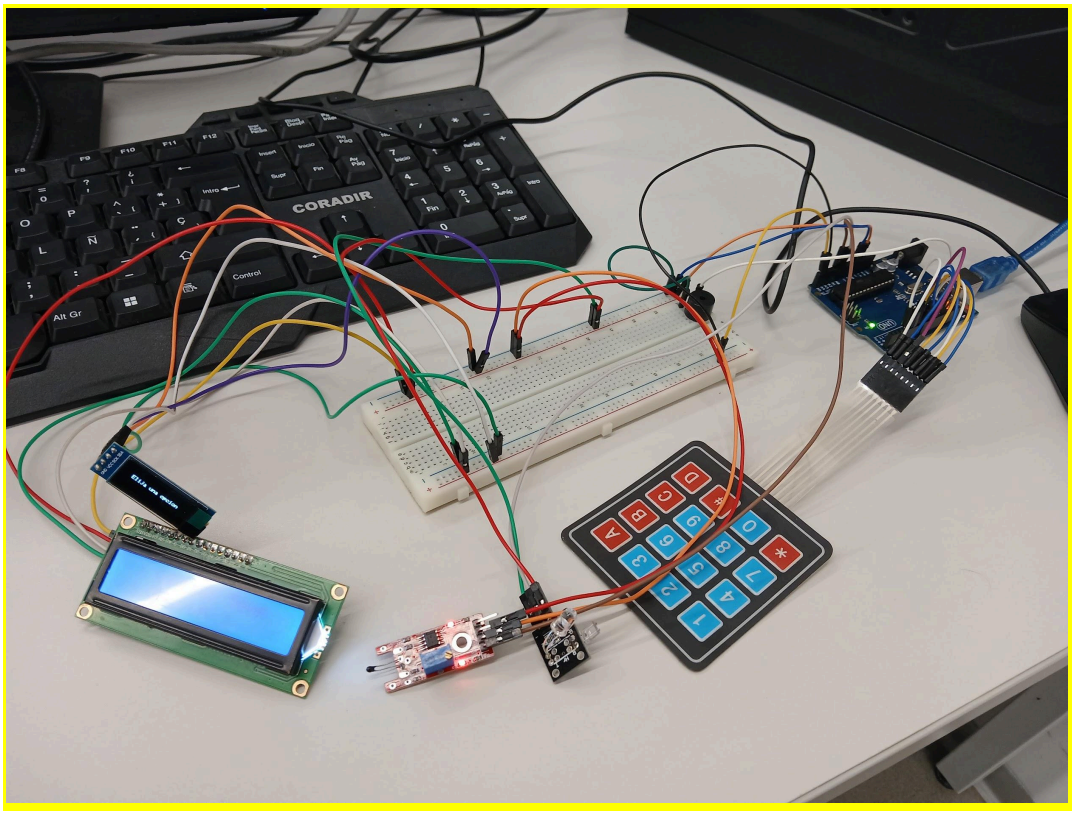
> lastPrint: guarda el momento de la última impresión.

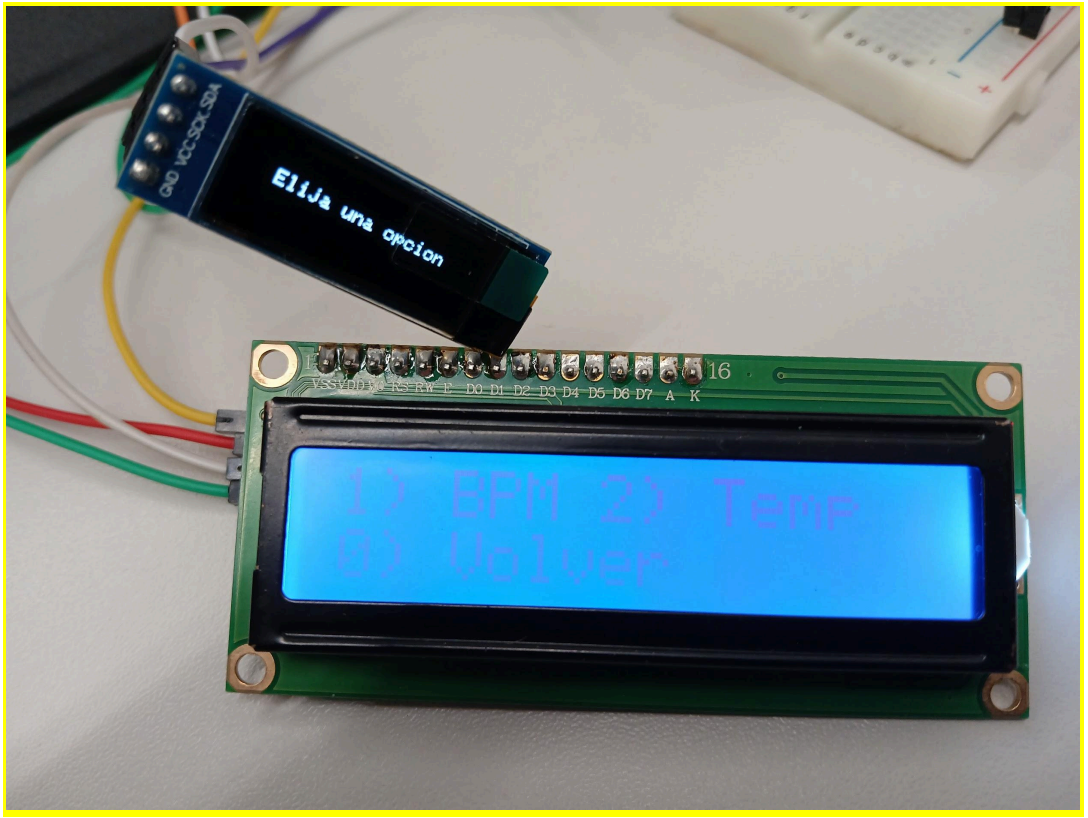
> interval: define el intervalo de actualización (por ejemplo, 3000 ms = 3 segundos).

Función dentro del sistema:

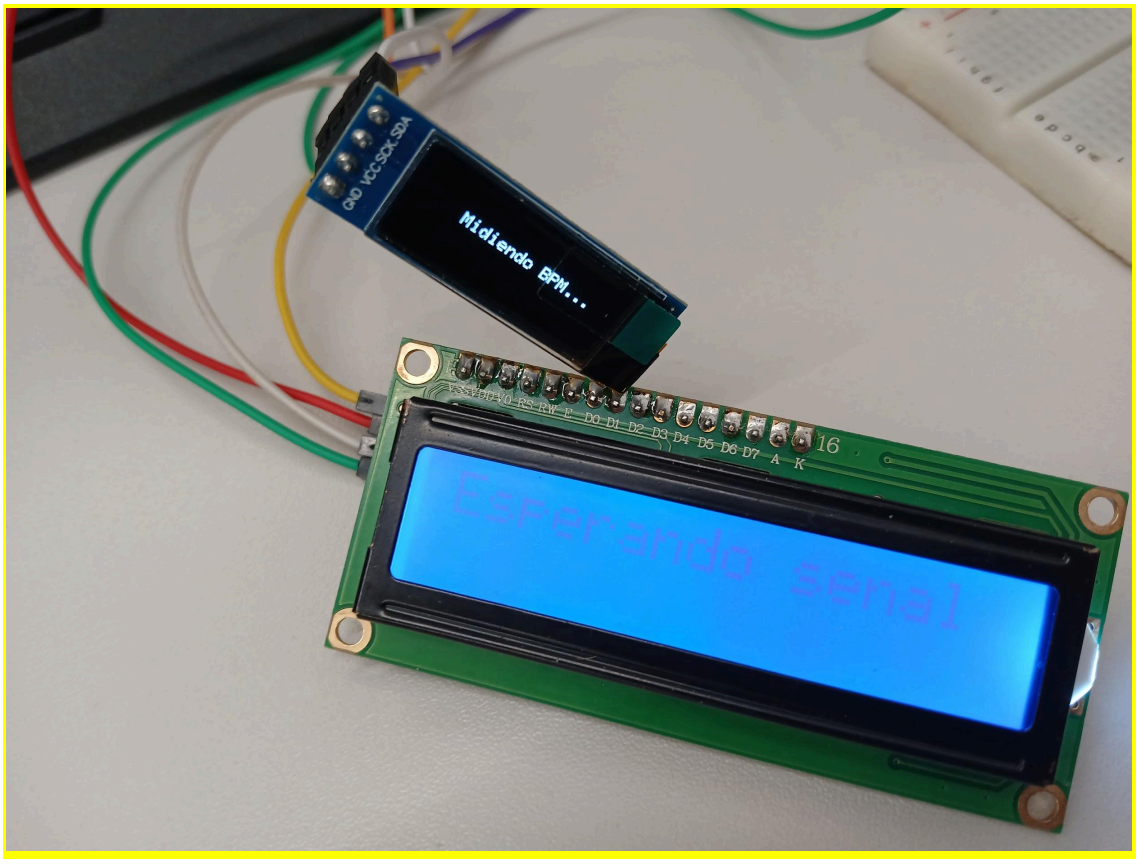
El control por tiempo evita que el valor del BPM cambie demasiado rápido en pantalla y mantiene una lectura estable.

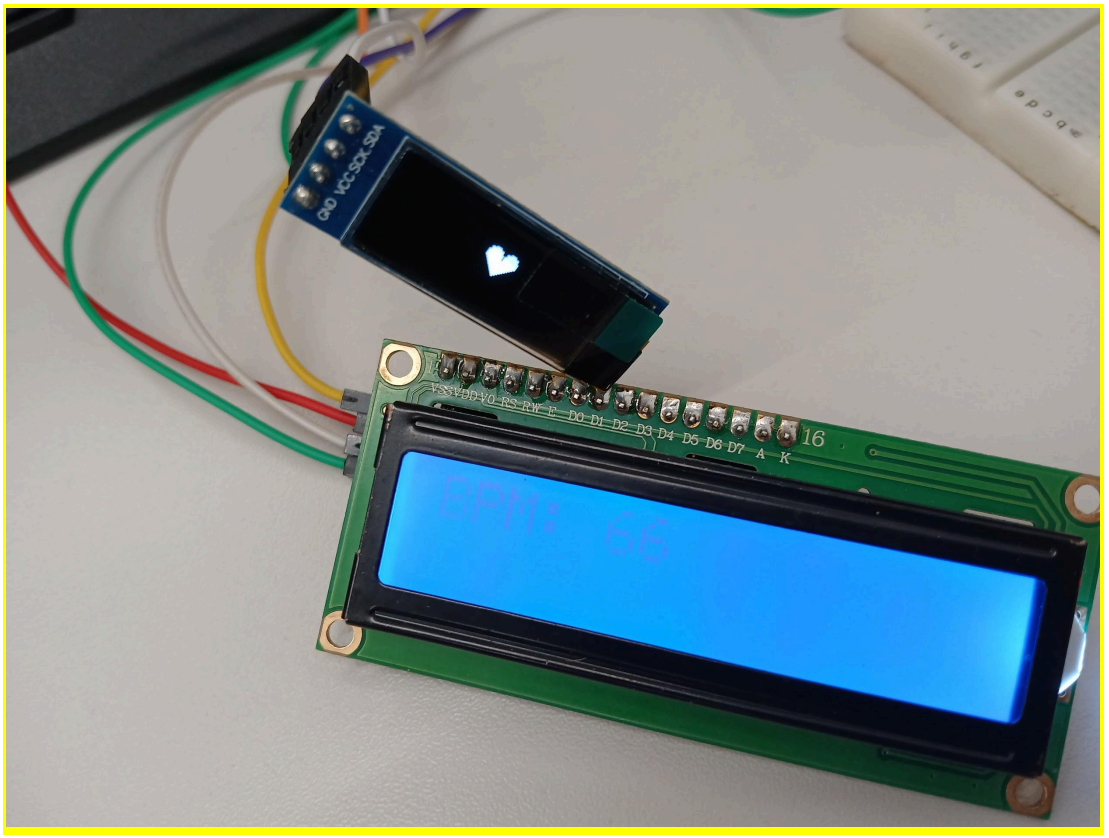
VI. CAPTURAS DEL SISTEMA FUNCIONANDO






GND VCC SCK SDA
Temp:
31.2 C
16
D0 D1 D2 D3 D4 D5 D6 D7 A K
Temp Prom:
31.18 C


GND VCC SCK SDA
Midiendo BPM...
16
D0 D1 D2 D3 D4 D5 D6 D7 A K
Esperando señal



VII. CONCLUSIONES DEL GRUPO

A lo largo del desarrollo de este proyecto, logramos integrar de manera efectiva conocimientos de electrónica y programación aplicados al uso del microcontrolador Arduino Uno. Se consiguió implementar un dispositivo capaz de medir el ritmo cardíaco (BPM) y la temperatura corporal, mostrando los resultados mediante pantallas LCD I2C y OLED, y permitiendo la interacción a través de un teclado matricial 4x4.

El uso de una máquina de estados permitió organizar el programa de forma modular y eficiente, garantizando un funcionamiento estable sin bloqueos. Además, la utilización de millis() para el control por tiempo y el método de detección de flancos para el conteo de pulsos demostraron ser herramientas adecuadas para obtener mediciones aproximadas.

Durante el proceso adquirimos y aplicamos habilidades para el manejo de señales analógicas y digitales, el uso de librerías externas en Arduino IDE y de la

importancia del diseño estructurado del software. También se reforzaron habilidades de trabajo en equipo, resolución de problemas y documentación técnica en github.

En conclusión, el proyecto cumplió con los objetivos planteados: diseñar, programar y poner en funcionamiento un sistema capaz de medir y visualizar variables fisiológicas básicas. Asimismo, nos permitió comprobar que el Arduino UNO es muy versátil para diferentes aplicaciones. El equipo ha quedado muy satisfecho con el resultado final, a pesar de las limitaciones en la precisión de los componentes utilizados.