Universidad Nacional de Quilmes
Departamento de Ciencia y Tecnología

Introducción a la programación de microcontroladores con Arduino

Proyecto final: Dispenser de agua con monitoreo web vía bluetooth

Alumnos:

- Ferro, Ignacio.
- Torboli, Nicolás

2026s1

Descripción del proyecto

El proyecto consiste en un dispenser de agua automático controlado por Arduino que detecta la presencia de un vaso mediante un sensor ultrasónico HC-SR04 y activa una bomba de agua a través de un relay cuando se detecta un objeto a menos de 10 cm de distancia. El sistema incorpora un sensor de nivel de líquido con boya magnética que bloquea automáticamente la bomba cuando el agua está baja, protegiendo al motor de trabajar en seco. Toda la información del sistema — distancia detectada, estado de la bomba, volumen de agua dispensado y alertas de nivel — se transmite en tiempo real de forma inalámbrica mediante un módulo Bluetooth HC-05 hacia una computadora, donde un servidor Python procesa los datos y los expone a través de WebSockets a una interfaz web visual accesible desde el navegador.

Materiales necesarios

Electrónica principal:

- Arduino Uno
- Sensor ultrasónico HC-SR04
- Módulo relay de 1 canal
- Bomba de agua DC 5V
- Módulo Bluetooth HC-05
- Sensor de nivel de líquido con boya magnética

Alimentación y protección:

- Fuente de alimentación externa 5V 2A (10W) regulada
- Capacitor electrolítico 470µF 16V (aprox. 5 unidades)
- Capacitor cerámico 100nF (código 104) (aprox. 5 unidades)
- Diodo 1N4007 (aprox. 5 unidades)

Divisor de voltaje para el HC-05:

- Resistencia 1kΩ × 1
- Resistencia 2kΩ × 1 (o dos resistencias de 1kΩ en serie)

Infraestructura:

- Breadboard
- Cables jumper macho-macho y macho-hembra (varios)

Requisitos para ejecutar el software

- Python 3.x instalado
- Arduino IDE instalado
- Navegador moderno con soporte WebSocket (Google Chrome, Microsoft Edge, Brave)

Librerías utilizadas

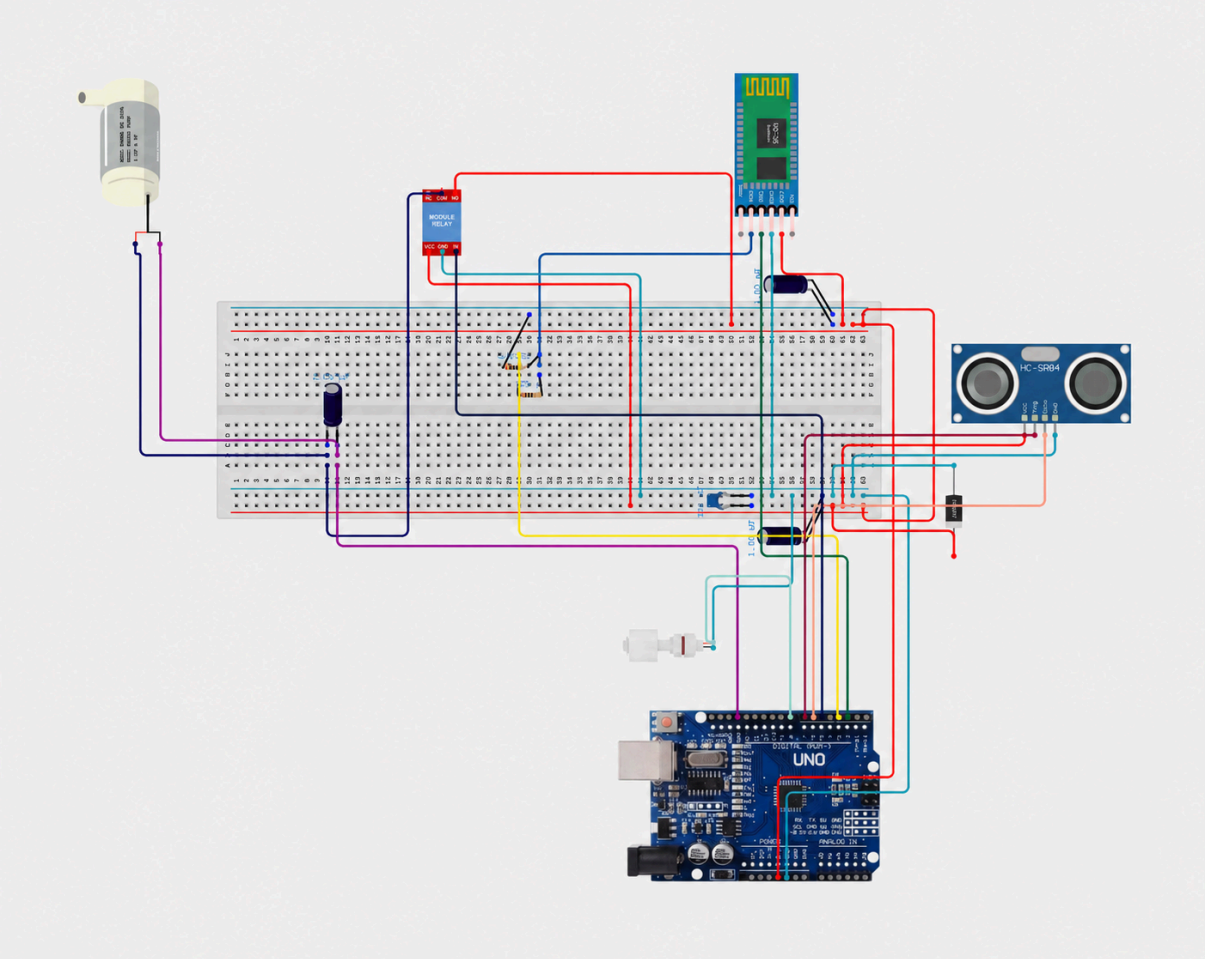
Arduino IDE:

- `SoftwareSerial.h`: incluida por defecto en Arduino IDE, no requiere instalación. Permite crear un puerto serie por software en los pines 2 y 3 para la comunicación con el HC-05.

Python:

- `pyserial`: lectura del puerto COM Bluetooth donde el HC-05 envía los datos del Arduino.
- `websockets`: servidor WebSocket que retransmite los datos en tiempo real al navegador.

Esquema



Paso a paso

1. Clonar el repositorio

Repositorio del proyecto:
https://github.com/ignacioferro80/dispenser-arduino

En la terminal de Git Bash ejecutar:

```
git clone
https://github.com/ignacioferro80/dispenser-arduino.git
cd dispenser-arduino
```

2. Instalar las dependencias de Python

En la terminal de Git Bash ejecutar:

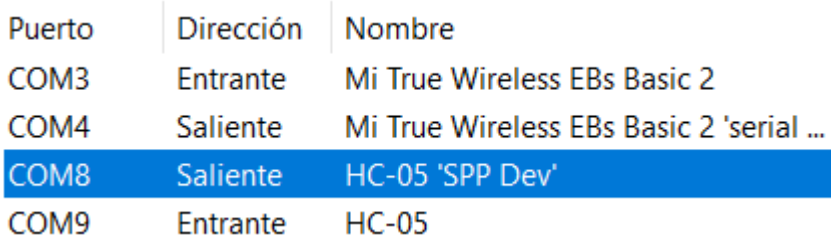
```
pip install pyserial websockets
```

3. Subir el código al Arduino UNO

- Abrir el archivo `codigo_arduino.ino` en el Arduino IDE
- Conectar la placa Arduino al equipo con su respectivo cable USB
- **Importante:** antes de subir el código, desconectar los pines 2 y 3 del HC-05 del breadboard — si están conectados durante la carga, SoftwareSerial interfiere con el proceso y la subida puede fallar
- Seleccionar la placa correcta en Herramientas → Placa → Arduino Uno
- Compilar y correr el código
- Una vez subido, volver a conectar los pines 2 y 3 del HC-05

4. Emparejar el HC-05 con la computadora

- Encender el Arduino mediante una corriente externa con el HC-05 conectado (el LED del módulo debería parpadear rápidamente)
- En Windows (11), abrir ***Configuración → Bluetooth → Agregar dispositivo***
- Seleccionar el HC-05 de la lista y usar el PIN 1234 cuando lo solicite
- Una vez emparejado, abrir el ***Panel de Control → Dispositivos e impresoras → Dispositivos → Más opciones de configuración de Bluetooth → puertos COM*** y buscar la entrada con dirección **Saliente** correspondiente al HC-05. Ese es el puerto que se utilizará en el servidor



Puerto	Dirección	Nombre
COM3	Entrante	Mi True Wireless EBs Basic 2
COM4	Saliente	Mi True Wireless EBs Basic 2 'serial ...
COM8	Saliente	HC-05 'SPP Dev'
COM9	Entrante	HC-05

ACLARACIÓN: Cada equipo puede tener un COM diferente. Por eso es esencial este paso.

5. Configurar el puerto COM en el servidor Python

- Abrí el archivo `servidor.py`
- Localizar esta línea cerca del principio del archivo:

```
PUERTO_BLUETOOTH = 'INSERTAR_PUERTO_COM'
```

- Reemplazar `INSERTAR_PUERTO_COM` por el puerto **saliente** del HC-05 identificado en el paso anterior. Por ejemplo, si el puerto saliente es COM5:

```
PUERTO_BLUETOOTH = 'COM5'
```

6. Correr el servidor Python

- Con el Arduino encendido y el HC-05 parpadeando, ejecutar en la terminal estando ubicado en la raíz del repositorio:

```
python servidor.py
```

- Se debería ver en la terminal:

```
Servidor WebSocket corriendo en ws://localhost:8765
Conectado al puerto COMX
```

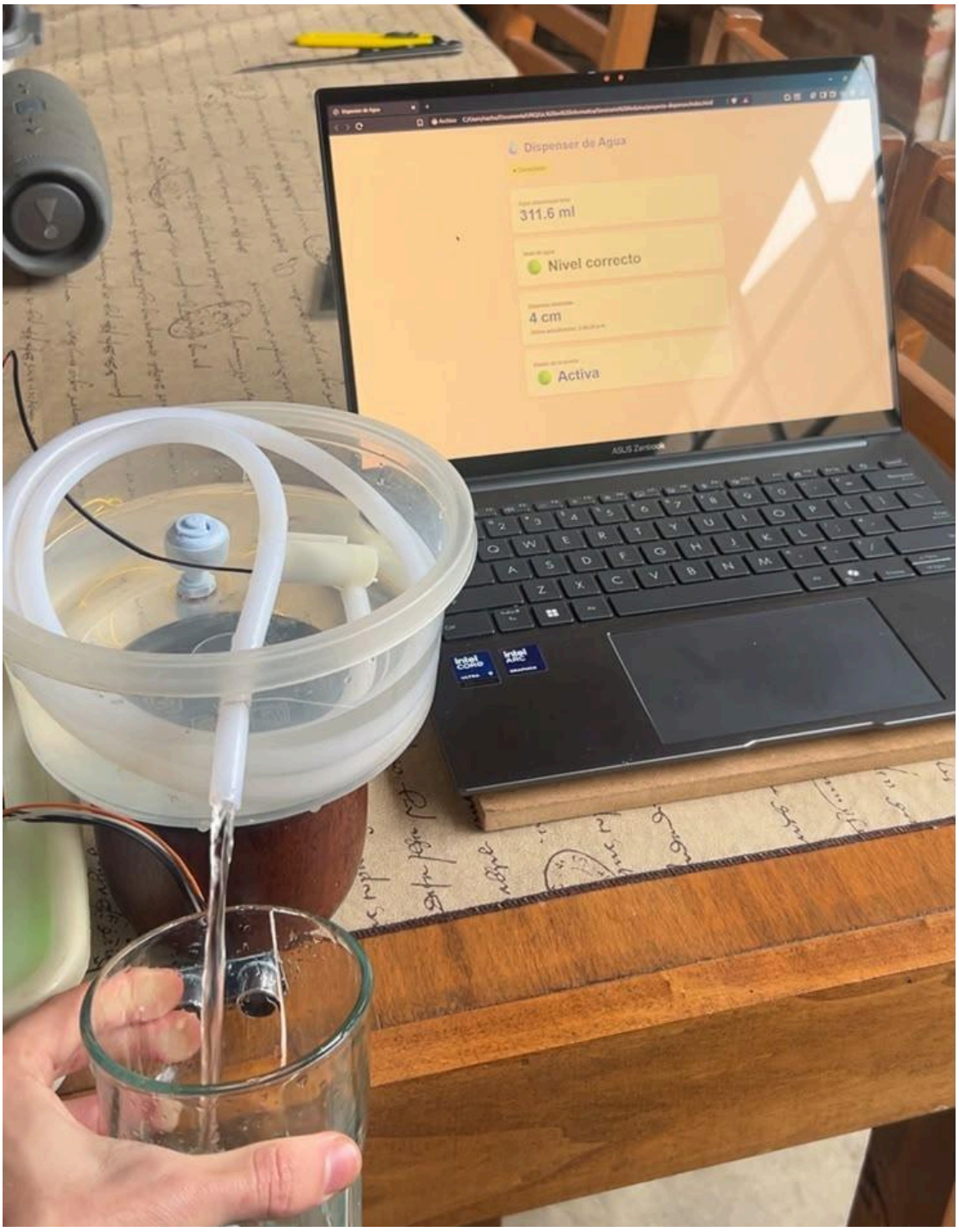
- Si se ve `Esperando conexion en COMX...` repetidamente, verificar que el HC-05 esté emparejado correctamente y que el número de puerto sea el saliente.

7. Abrir la interfaz web

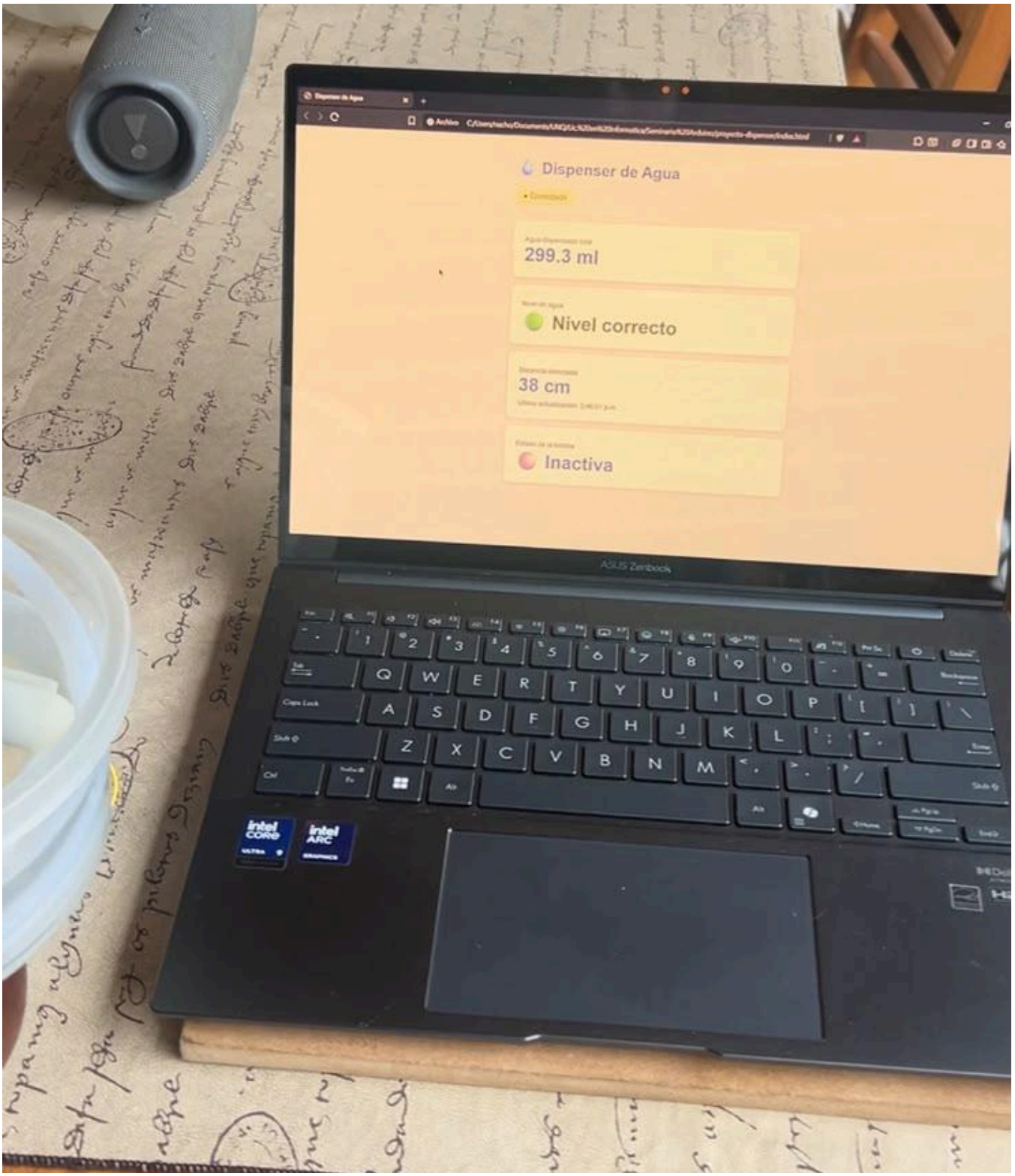
- Abrir el archivo `index.html` directamente en el navegador
- El indicador superior debería mostrar: ● Conectado
- Los datos del dispenser comenzarán a aparecer en tiempo real: distancia detectada, si el agua está a un nivel correcto, distancia hacia el objeto más cercano y el estado de la bomba.

8. Verificar el funcionamiento completo

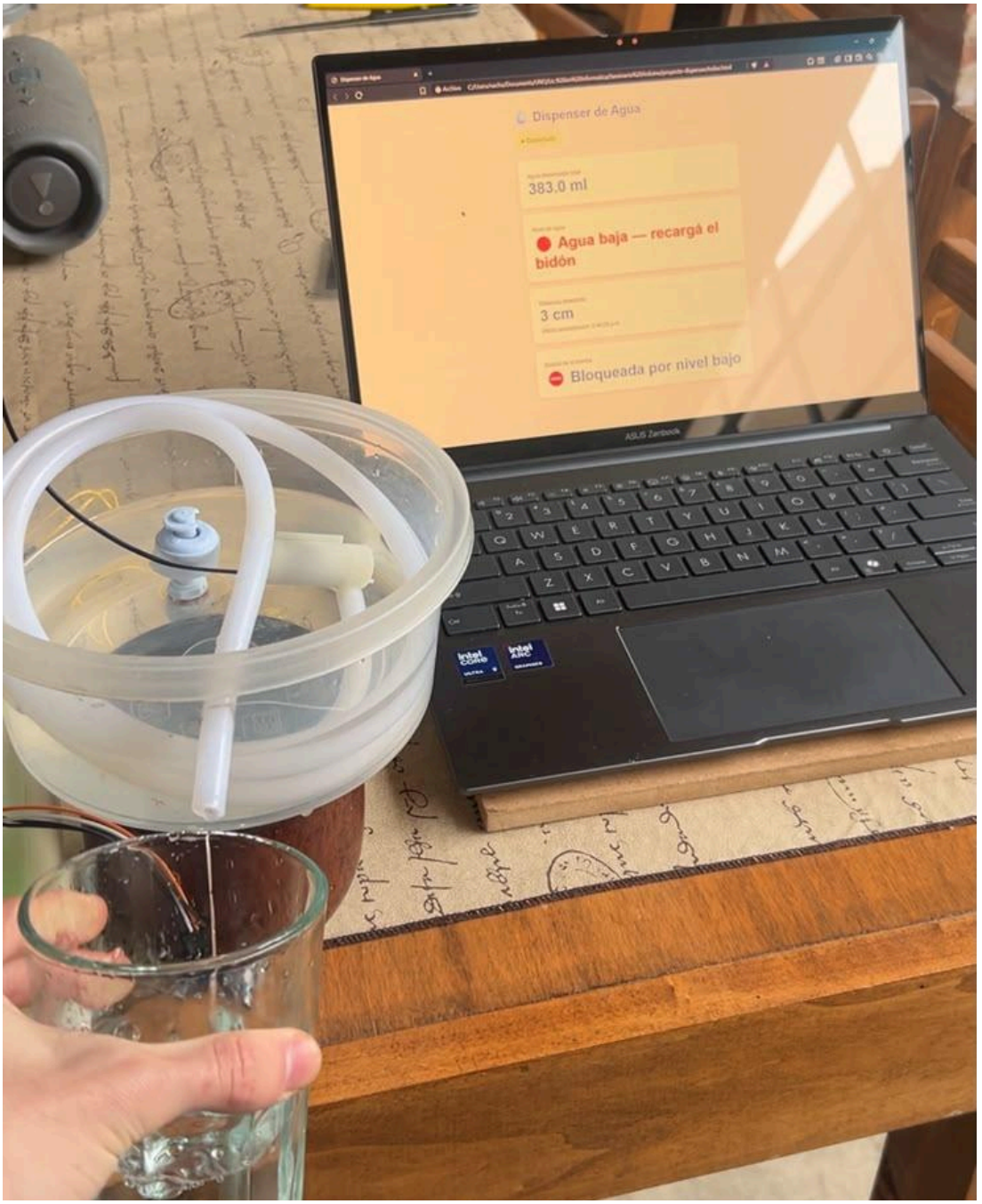
- Acercar un vaso a menos de 10 cm del sensor HC-SR04 → la bomba debería activarse, el volumen de agua dispensado aumentar mientras el dispenser esté activo, y el estado de la bomba decir "🟢 Activa"



- Al alejar el vaso → la bomba se detiene y el estado pasa a “⚪Inactiva”



- Al bajar la boya del sensor de nivel hasta que toque el imán → el sistema bloquea la bomba y muestra "🔴 Agua baja — recargá el bidón" en la interfaz



Complicaciones durante la ejecución

Comportamiento inconsistente de la bomba

Durante el armado del circuito, se detectó que el cable rojo (positivo) de la bomba de agua presentaba un deshilachado en su conexión, lo que generaba un inconsistente funcionamiento de la bomba.

Solución implementada:

Se utilizó una pinza para unir y ajustar firmemente los hilos de cobre expuestos del cable, asegurando un contacto estable y confiable en la conexión.

Interferencia eléctrica y reseteo del Arduino

Durante las pruebas de funcionamiento, se detectó que la aplicación desarrollada en Python (que recibía datos correctamente vía Bluetooth) comenzó a mostrar cortes de conexión cada vez que la bomba se activaba, pese a que el sistema seguía funcionando correctamente de fondo (sensor y relé operando con normalidad).

Se identificó que esto se debía a que el arranque del motor de la bomba generaba un pico de corriente que, al compartir el mismo riel de alimentación con el resto de los componentes, hacía caer momentáneamente la tensión de todo el circuito. Esto provocaba:

- Desconexiones de la comunicación Bluetooth con la aplicación en Python (conectando automáticamente luego de unos segundos).
- El LED de encendido del Arduino se apagaba momentáneamente, confirmando un reinicio completo del microcontrolador.

Solución implementada:

Moviendo la alimentación de 5V y el terminal NO del relé hacia el otro riel positivo de la breadboard

Limitación de acceso remoto vía Bluetooth (arquitectura de la página web)

Se planteó como objetivo poder visualizar los datos del Arduino desde una página web accesible por link, desde cualquier dispositivo (celular o PC), con el Arduino alimentado de forma independiente (sin depender de una PC conectada por cable).

Se identificó una limitación fundamental: el Bluetooth tiene un alcance físico limitado , por lo que no puede transmitir datos directamente a una página web alojada en internet — se requiere necesariamente un dispositivo intermedio (PC o similar) físicamente cerca del Arduino, que reciba los datos por Bluetooth y los reenvía hacia internet.

Solución implementada:

- Se desarrolló un script en Python (utilizando la librería pyserial) que se ejecuta en la PC ubicada cerca del Arduino, actuando como puente: lee los datos recibidos por el puerto COM asociado al Bluetooth y los reenvía a un servicio en la nube.
- Desde ese servicio, la página web consulta y muestra los datos en tiempo real, permitiendo el acceso desde cualquier dispositivo con conexión a internet, sin necesidad de que el usuario final tenga Bluetooth ni esté cerca del Arduino.

Esta arquitectura resuelve el requerimiento de acceso remoto, aunque mantiene la dependencia de que la PC puente permanezca encendida y conectada a internet para que los datos se sigan actualizando.

Problema de identificación de puertos Bluetooth (COM ocultos)

Al intentar identificar el puerto COM correspondiente al módulo Bluetooth ya emparejado, se detectó que los puertos no aparecían visibles o accesibles correctamente en el Administrador de Dispositivos de Windows, dificultando la conexión desde la aplicación en Python.

Solución implementada:

Reiniciar la PC permitió que Windows regenerara correctamente la lista de puertos COM asociados al dispositivo Bluetooth emparejado, haciéndolos visibles y utilizables para establecer la conexión.