UNIVERSIDAD DE TARAPACÁ
Universidad del Estado

Ingeniería
Computación e Informática

PLAN DE PROYECTO FASE 3

Claw-tylda

Profesor:

Baris Klobertanz.

Integrantes:

Juan-Daniel Castillo.
Javier Echeverria.
Alexander Pinto.
José Terrazas.

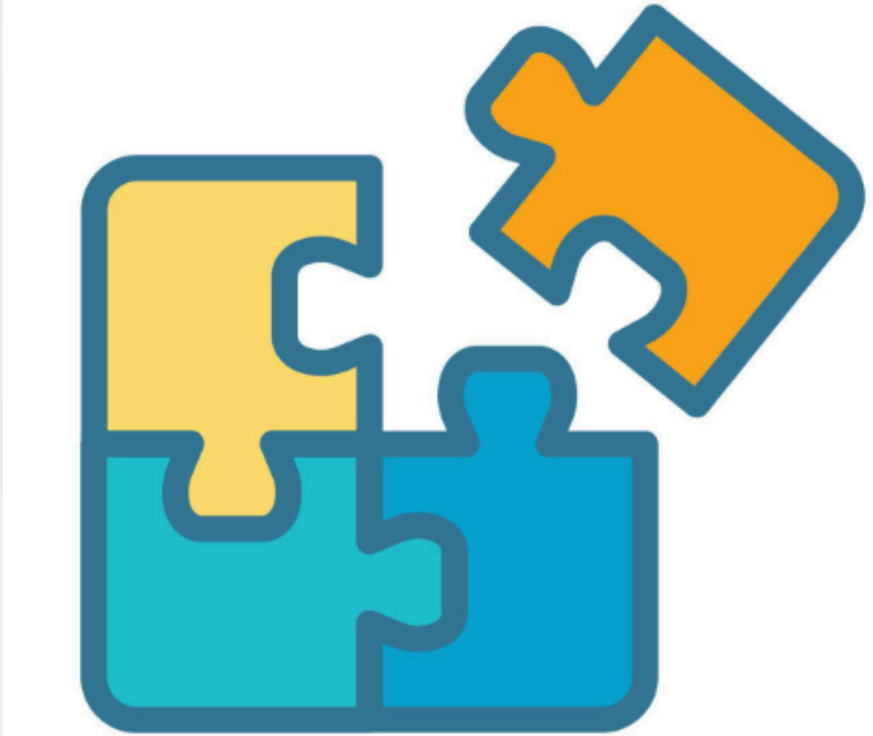
Índice de Contenidos

PROBLEMA

- Falta de optimización en el proceso de carga de materiales fragmentados en la industria minera de Chile.

- **Impacto:** La industria minera representa un 11,7% del PIB de Chile.

- **Solución:** Minería 4.0.



OBJETIVO GENERAL

Implementar una garra robótica capaz de cargar material minero.

Eficiencia

Trabajadores

Empresas

OBJETIVOS ESPECÍFICOS

OBJETIVO	ACTIVIDAD	TIEMPO
Investigar diseños para el robot	Analizar al menos tres diseños, evaluando resistencia, movilidad y tamaño	Primeras 2 semanas
Investigar bibliotecas compatibles con Lego Spike Prime	Investigar al menos una librería compatible	Primeras 2 semanas
Ensamblar diseño funcional	Estructura que permita abrir/cerrar garra, elevarse y desplazarse horizontalmente	Primer mes
Codificar movimientos	Apertura/cierre, ajuste de altura y desplazamiento horizontal	Hasta segundo mes y dos semanas
Pruebas de compatibilidad	Comprobar que la codificación y la estructura funcionen en conjunto	2 semanas desde tener ensamblado un prototipo funcional

OBJETIVOS ESPECÍFICOS

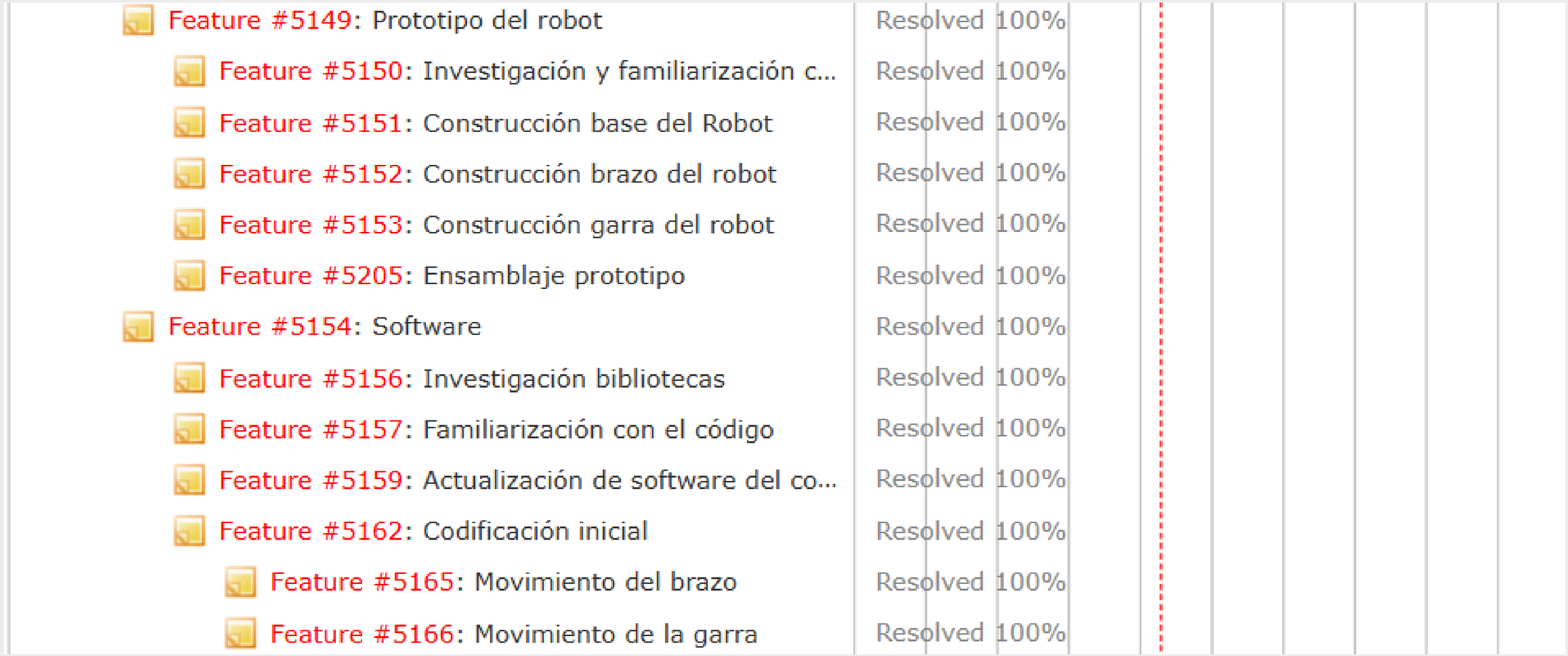
OBJETIVO	ACTIVIDAD	TIEMPO
Definir diseño final	Ajustar problemas encontrados durante las pruebas de compatibilidad	Durante el segundo mes de proyecto
Ajustes de codificación	Ajustar la codificación para que los movimientos no rompan la estructura física	Dos semanas antes de la finalización del proyecto
Selección de bibliotecas para interfaz	Investigar y seleccionar librerías para implementar la interfaz gráfica	Primer mes y medio
Implementar interfaz gráfica	Desarrollar una interfaz gráfica que permita al usuario controlar los movimientos del robot	Al menos 3 semanas antes de la finalización del proyecto

ESTRUCTURA ORGANIZACIONAL

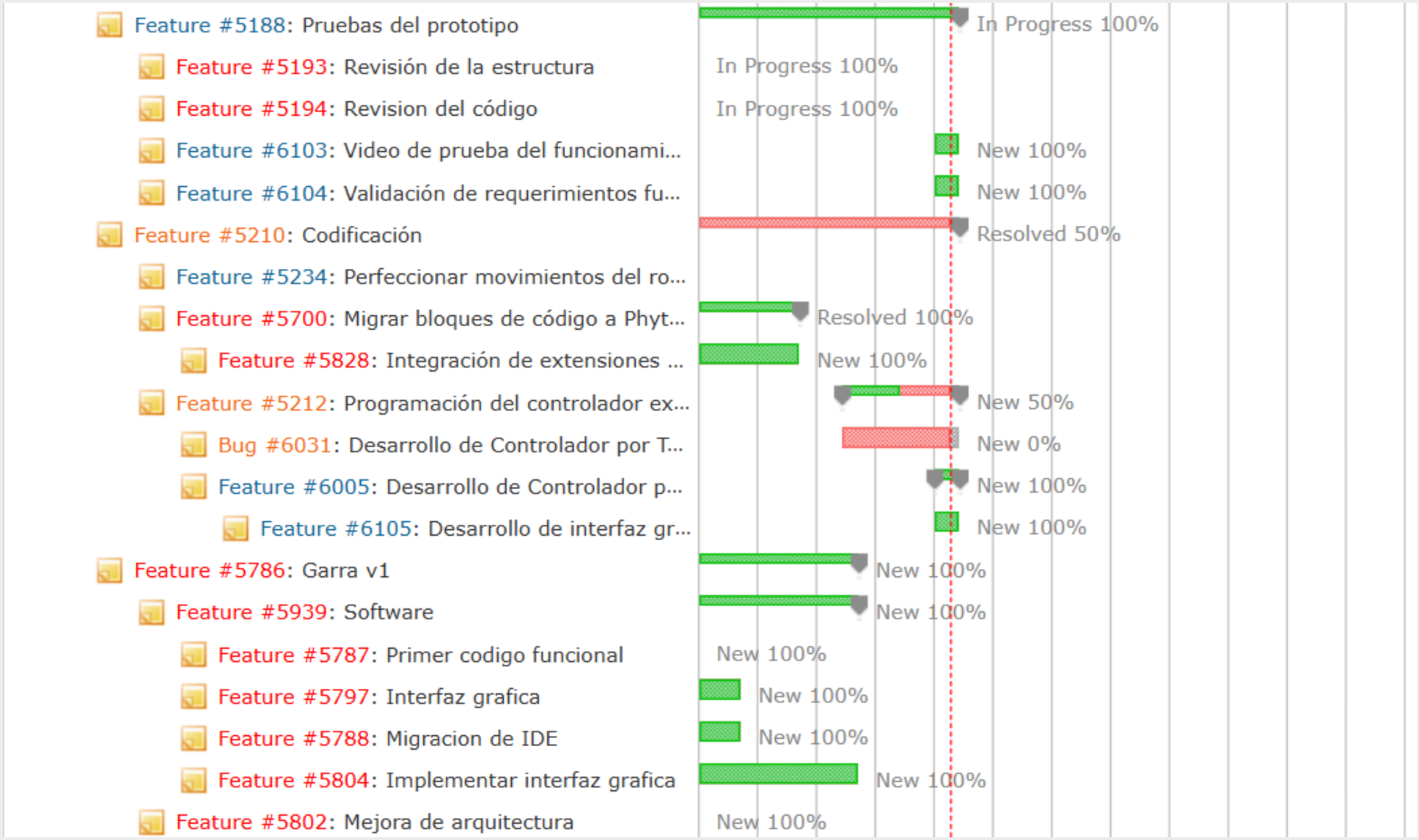
ROL	RESPONSABLE
Jefe de proyecto	Javier Echeverria
Programador	Juan-Daniel Castillo
Ensamblador	Alexander Pinto
Documentador	José Terrazas

Carta Gantt

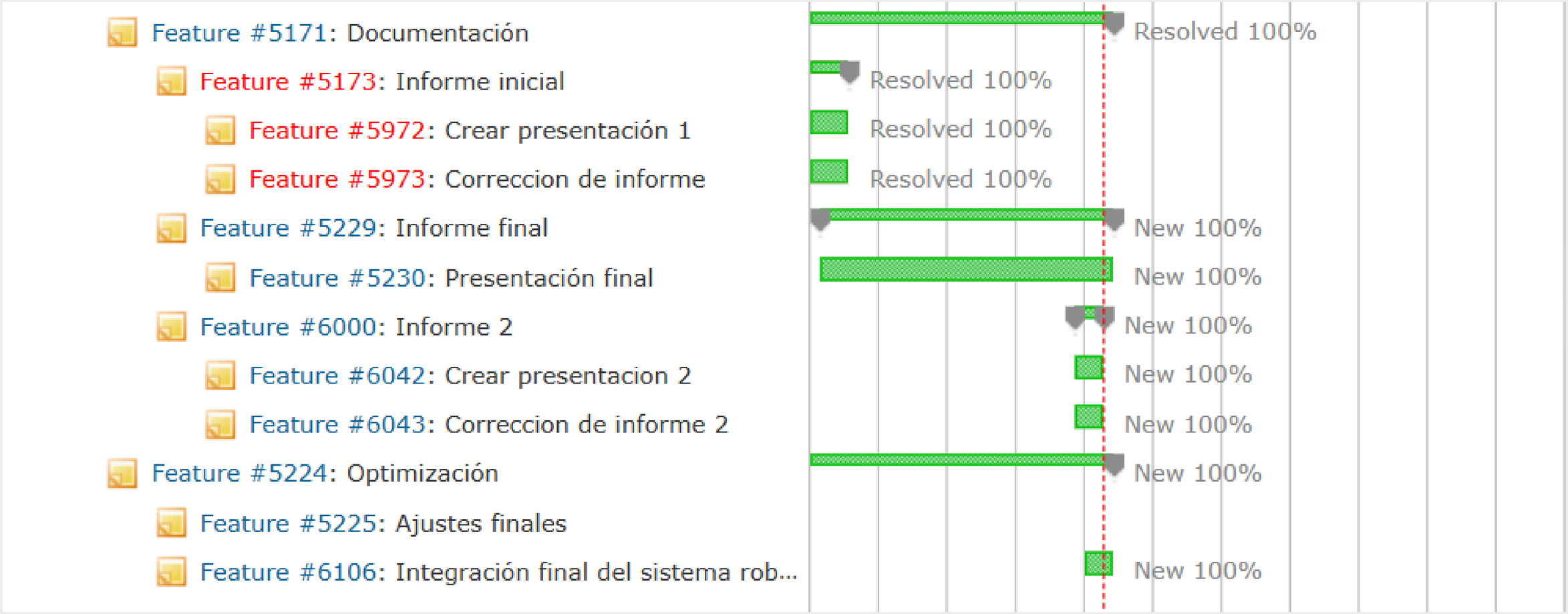
Inicio del proyecto



Desarrollo y pruebas



Cierre del proyecto



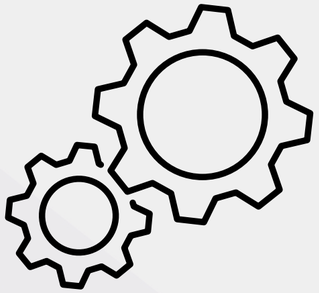
Gestion de riesgos

Riesgos	Nivel de impacto	Acción remedial
Abandono de personal	1	Reestructurar la gestión de tareas y recortar labores con menor importancia, para no sacrificar la eficacia del proyecto.
Pérdida de robot	1	Comprar un nuevo kit de lego spike prime con el dinero en conjunto del grupo y poner una nueva custodia al robot.
Atraso en el cumplimiento de tareas	2	Instaurar límites de tiempo y organizar las tareas parecidas para mejorar la eficiencia y reajustar los horarios para encontrar mayor tiempo en los encargos críticos en caso de atraso.

Riesgos	Nivel de impacto	Acción remedial
Ausencia repentina por fuerza mayor	2	Llenar ese rol con un integrante que tenga las capacidades que se requieren o retrasar tareas que se puedan postergar.
Rotura de piezas	3	Solicitar un reemplazo de las piezas rotas o comprar nuevas.
Inestabilidad del diseño del robot	4	Investigar un prototipo de mayor estabilidad o hacerle ajustes al mismo diseñado y repartir el peso a los puntos que provocan las inestabilidad.
Fallo en la programación del robot	3	Depurar el código para enmendar el fallo.

Riesgos	Nivel de impacto	Acción remedial
Problemas con la señal de internet	5	Probar con otras señales de la universidad, cambiar a una conexión privada o por vía cable ethernet.

FUNDAMENTOS DE LOS MOVIMIENTOS



TORQUE

FRICCION Y FUERZA DE GARRA

TORQUE DE BRAZO

- **Radio: 20 [cm] = 0,2 [m]**
- **Masa: 30 [g] = 0,03 [kg]**

- **Velocidad deseada:**
- **v = 0,3125 [m/s]**

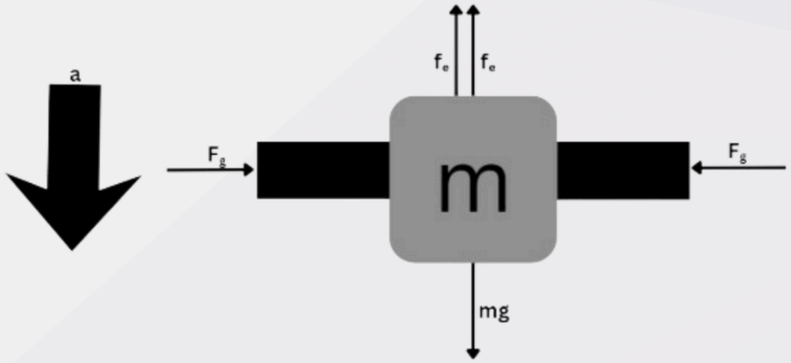
- **Aceleración deseada:**
- **a = 1,5625 [m/s^2]**

- **Fuerza: F = ma = 4,69 * 10^{-2} [N]**

- **Torque: T = Fr = 9,38 * 10^{-3} [N]**

FUERZA DE GARRA

- **Coeficiente de friccion: u = 0,6**



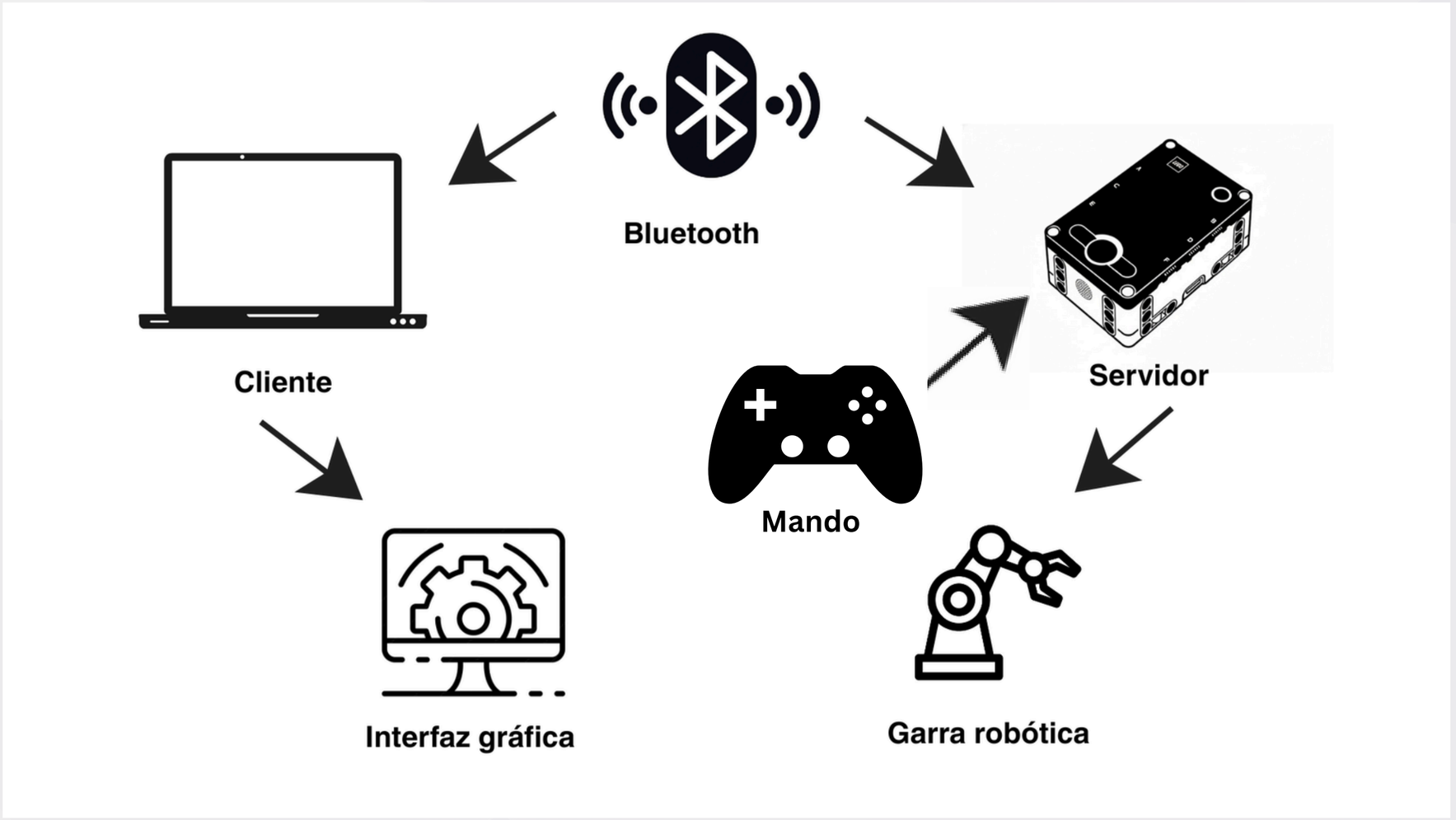
$\Sigma F = ma \longrightarrow f_e = m(g-a)/2$

$f_e < Nu \longrightarrow f_e < F_g u \longrightarrow m < 2F_g u/ga$

$m_{max} = 0,39F_{gT}$

$m_{max} = 0,78\ [kg]$

ARQUITECTURA DE SOFTWARE



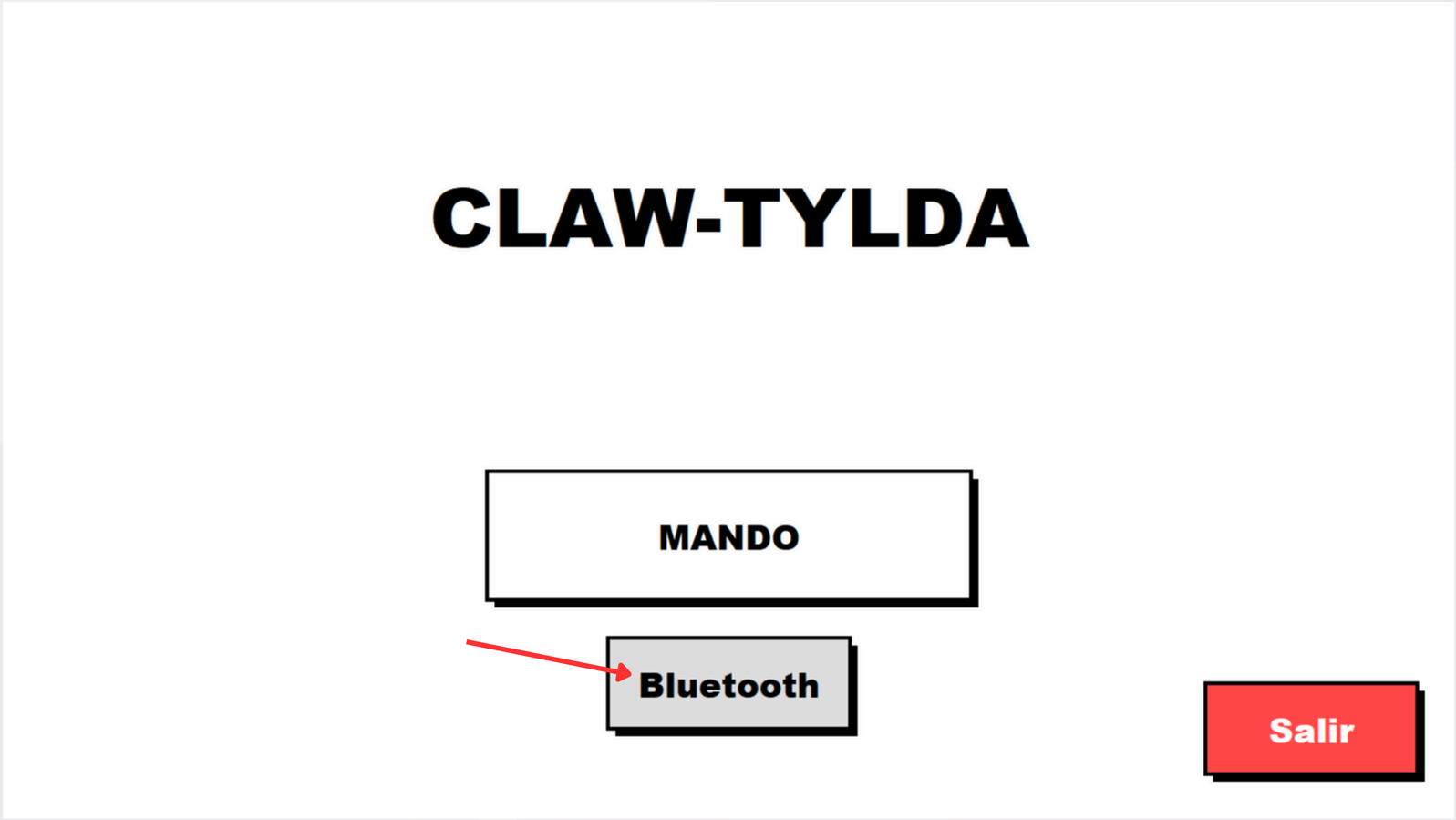
REQUERIMIENTOS FUNCIONALES

- El robot debe poder cerrar y abrir su garra.
- El robot debe poder ajustar la altura de su brazo.
- El robot debe poder mover su garra horizontalmente.
- El robot debe poder recoger un bloque de lego determinado usando su garra.
- El robot debe dejar este mismo bloque de lego en una base o plataforma de carga.
- El usuario debe poder controlar todos los movimientos del robot y finalizar su funcionamiento, a través de la interfaz gráfica.
- El sistema debe validar las entradas del usuario y notificar cualquier error, mediante la interfaz gráfica o el robot.

REQUERIMIENTOS NO FUNCIONALES

REQUERIMIENTO	DESCRIPCIÓN
Disponibilidad	El robot debe poder funcionar de manera continua durante una hora.
Rendimiento	La latencia máxima entre la acción del usuario y el movimiento del robot debe ser como máximo de 1 segundo.
Usabilidad	La interfaz gráfica debe ser intuitiva y fácil de usar, además debe soportar conexión por bluetooth.
Robustez	La estructura del robot debe mantenerse estable durante los movimientos y la interfaz debe notificar si ocurre algún error.
Control	Los movimientos deben poder ser controlados por el usuario, al menos de una forma.

IMPLEMENTACIÓN
PASO 1



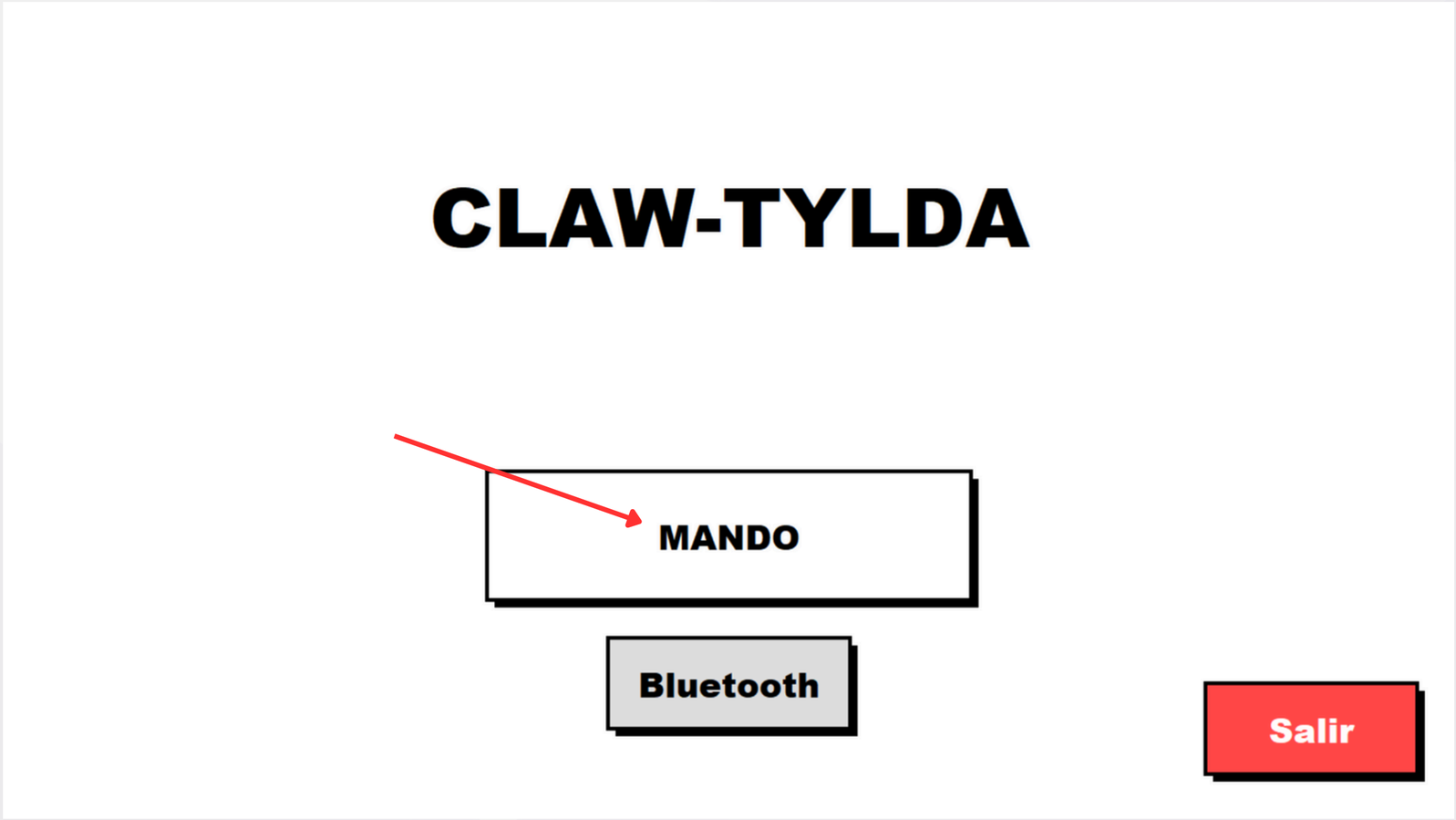
IMPLEMENTACIÓN

```
async def tarea():
    global hub, conectando
    try:
        hub = await conectar_hub()
        canvas.itemconfig(boton_bt_rect, fill="#4CAF50")
        ocultar_notificacion()
    except:
        canvas.itemconfig(boton_bt_rect, fill="#F44336")
    conectando = False

asyncio.run_coroutine_threadsafe(tarea(), loop)
```

IMPLEMENTACIÓN

```python
async def conectar_hub():
    dispositivo_ble = await find_device(name="Sp-11")
    if not dispositivo_ble:
        print("No se pudo encontrar al Hub Sp-11")
        return None

    hub = PybricksHubBLE(dispositivo_ble)
    await hub.connect()
    print("Conexión al hub completa")
    return hub
```

IMPLEMENTACIÓN
PASO 2



IMPLEMENTACIÓN

```
def boton_mando_a():
    global hub

    if hub is None:
        async def conectar_y_enviar():
            global hub
            hub = await conectar_hub()
            if hub is not None:
                await enviar_programa(hub)
        asyncio.run_coroutine_threadsafe(conectar_y_enviar(), loop)
    else:

        asyncio.run_coroutine_threadsafe(enviar_programa(hub), loop)
    abrir_control()
```

IMPLEMENTACIÓN

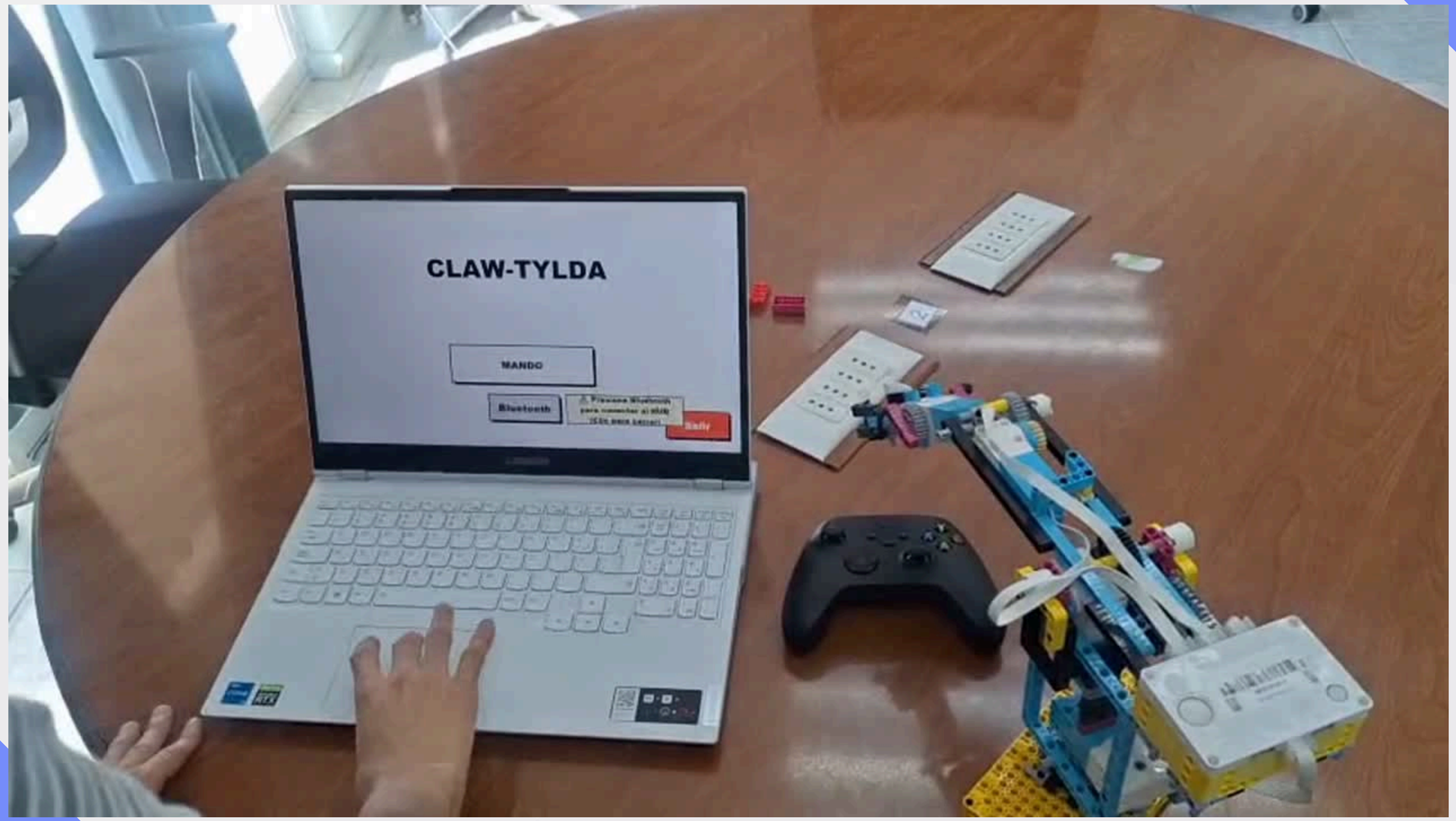
```python
async def enviar_programa(hub):
    programa_hub = generar_programa_hub()

    with tempfile.NamedTemporaryFile(delete=False, suffix=".py") as f:
        f.write(programa_hub.encode("utf-8"))
        ruta = f.name

    await hub.run(ruta)
    os.remove(ruta)
```

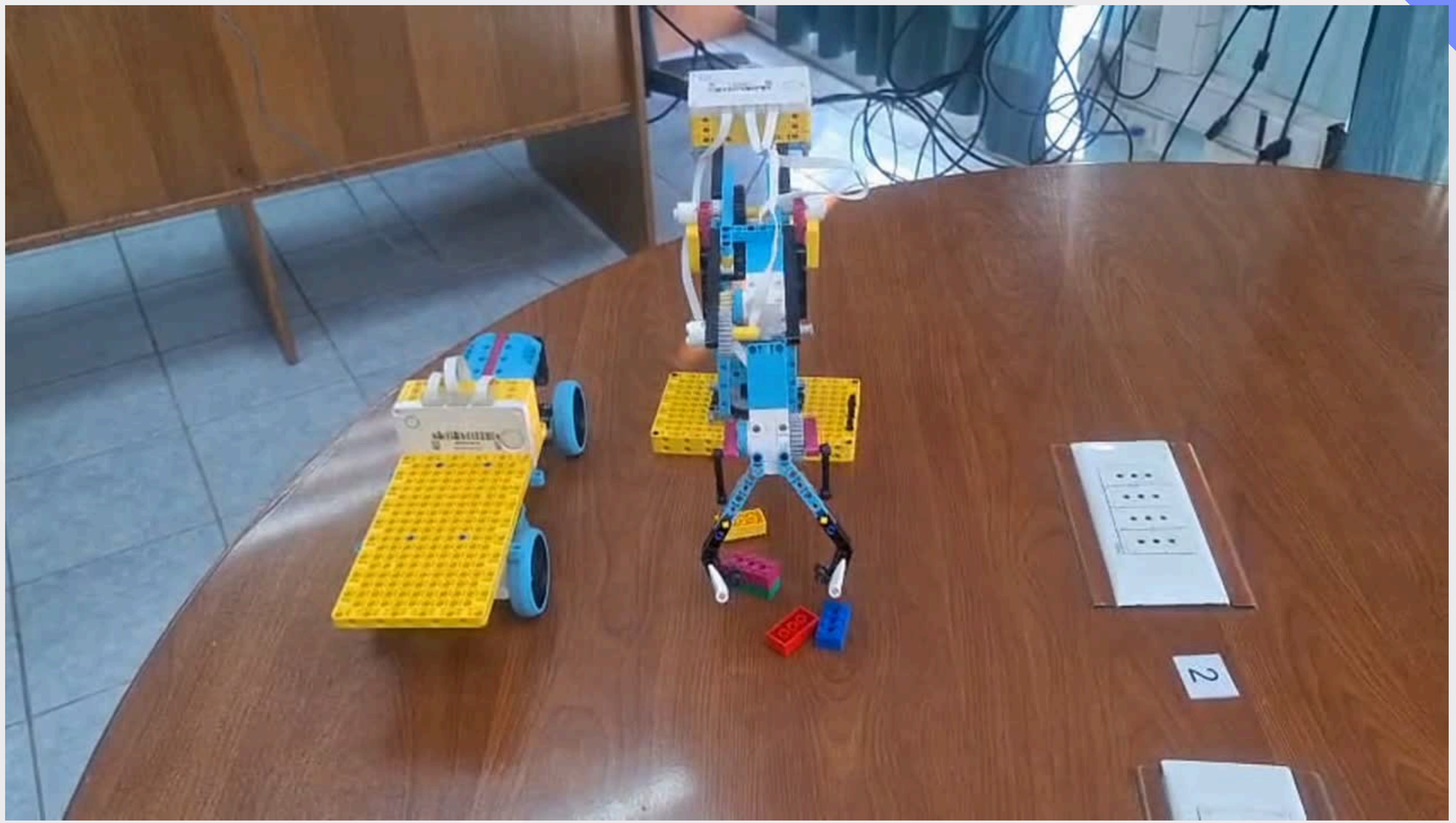
PANTALLA DE CONTROL

CONTROLES
MANDO

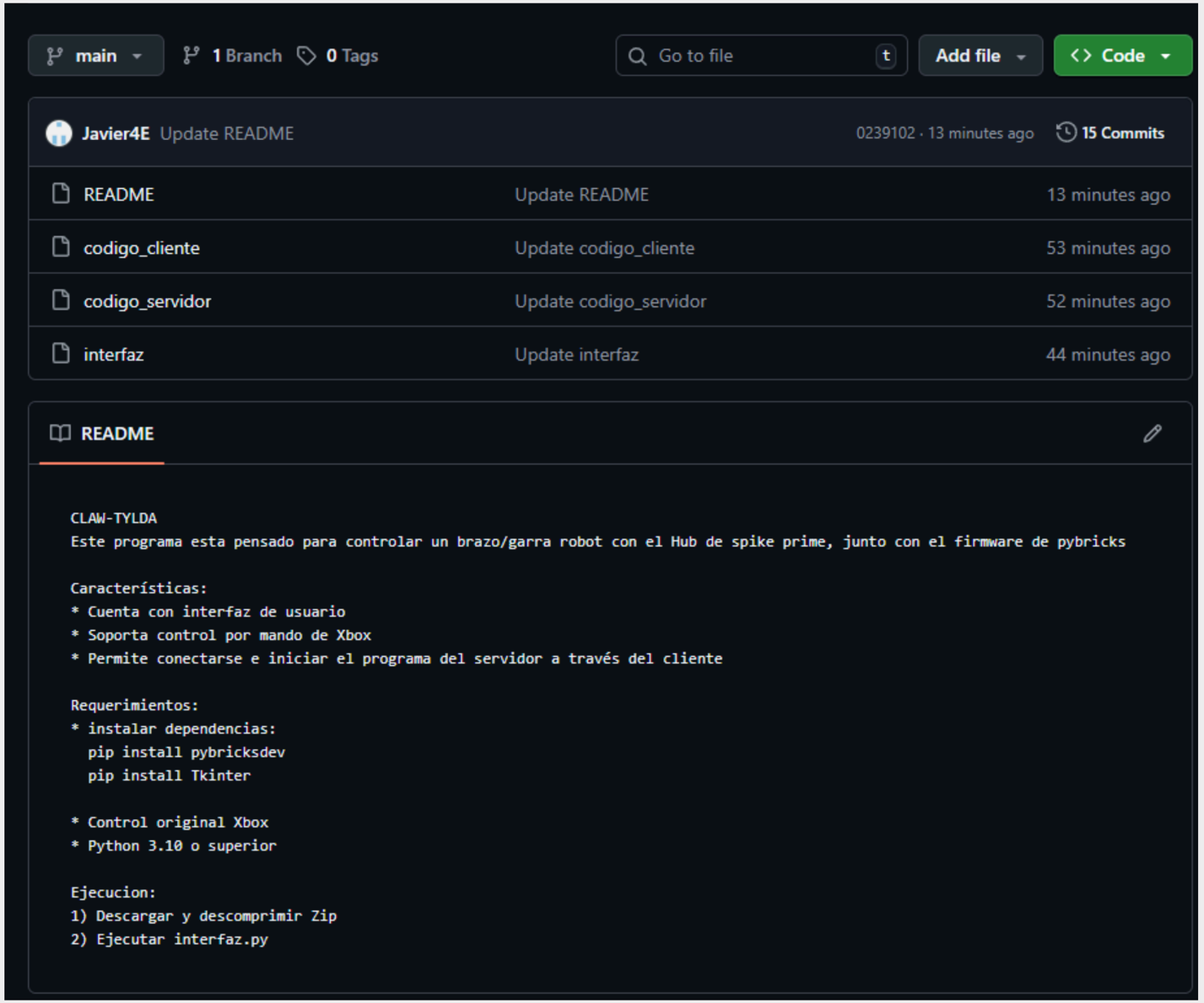
VOLVER

PRUEBA FUNCIONAL


CLAW-TYLDA
MANDO
Bluetooth


2

REPOSITORIO DE GITHUB



RELEASE

CLAW-TYLDA v1.0.0 (codigo fuente) Latest

Compare

Javier4E released this 4 minutes ago v1.0.0 0239102

Primera versión completamente funcional del proyecto CLAW-TYLDA.

Esta versión incluye:

- Código de interfaz gráfica (Tkinter)
- Código de cliente y servidor para su conexión por Bluetooth.
- Programa de Pybrick enviado directamente al Hub.

Assets 2

Source code (zip) 14 minutes ago

Source code (tar.gz) 14 minutes ago

CONCLUSIONES

MUCHAS
GRACIAS