



UNIVERSIDAD DE TARAPACÁ
Universidad del Estado

Ingeniería@
Computación e Informática

PLAN DE PROYECTO FASE 2

Claw-tylda

Profesor:

Baris Klobertanz.

Integrantes:

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

Índice de Contenidos

01 REQUERIMIENTOS

02 ARQUITECTURA DE SOFTWARE

03 FUNDAMENTOS DE LOS MOVIMIENTOS

04 DISEÑO DE LA INTERFAZ GRÁFICA

05 CLIENTE

06 SERVIDOR

07 ESTADO ACTUAL DEL PROYECTO

08 CONCLUSIÓN

REQUERIMIENTOS

- **Problema:** Falta de optimización en el proceso de carga de materiales fragmentados.
- **Cliente:** Empresas mineras.
- **Usuario:** Trabajadores encargados de transportar los minerales.



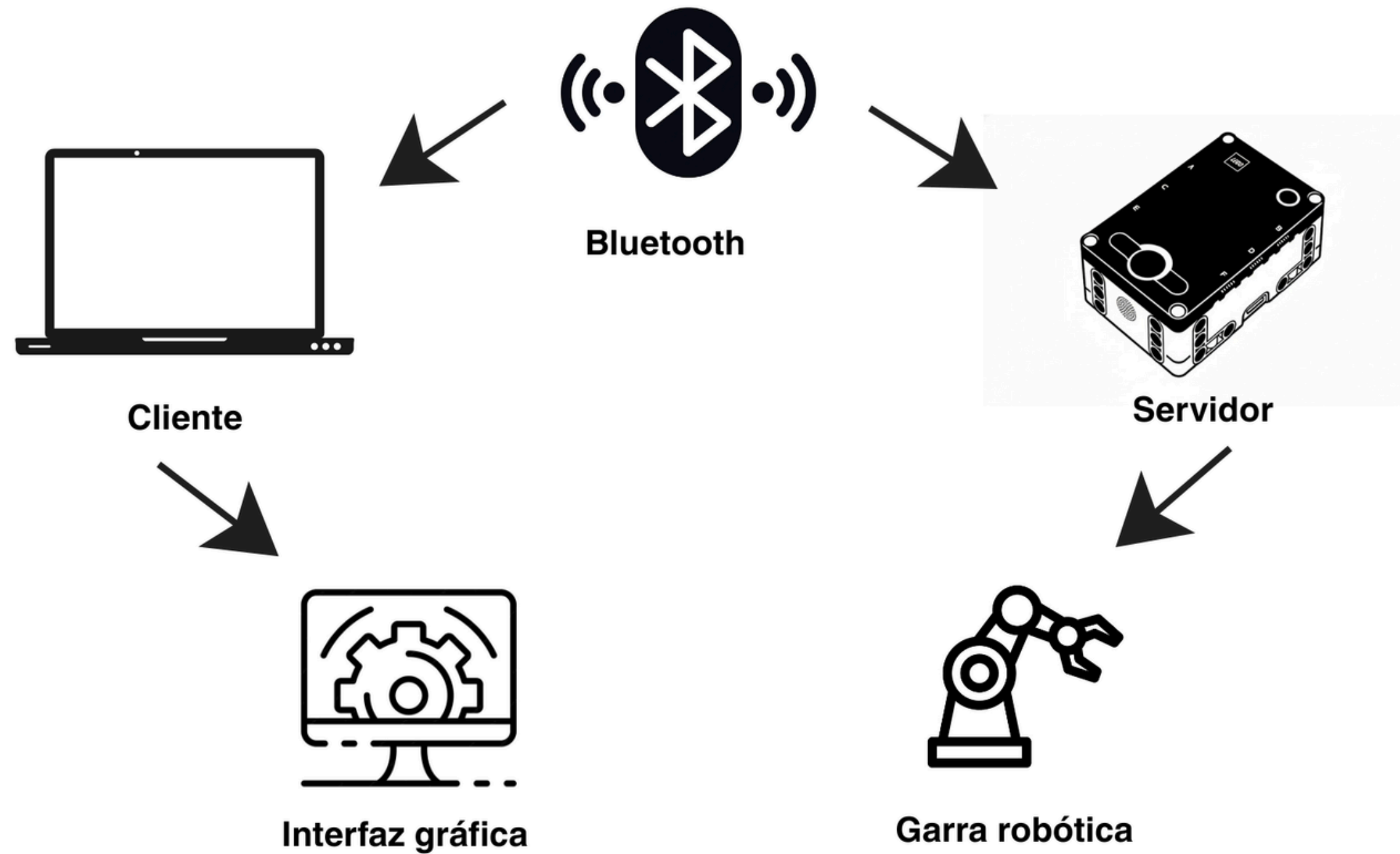
REQUERIMIENTOS FUNCIONALES

- 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.

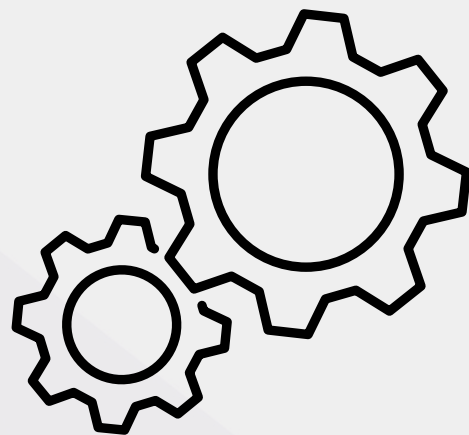
REQUERIMIENTOS NO FUNCIONALES

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.

ARQUITECTURA DE SOFTWARE



FUNDAMENTOS DE LOS MOVIMIENTOS



TORQUE



**FRICCION Y FUERZA
DE GARRA**

TORQUE DE BRAZO

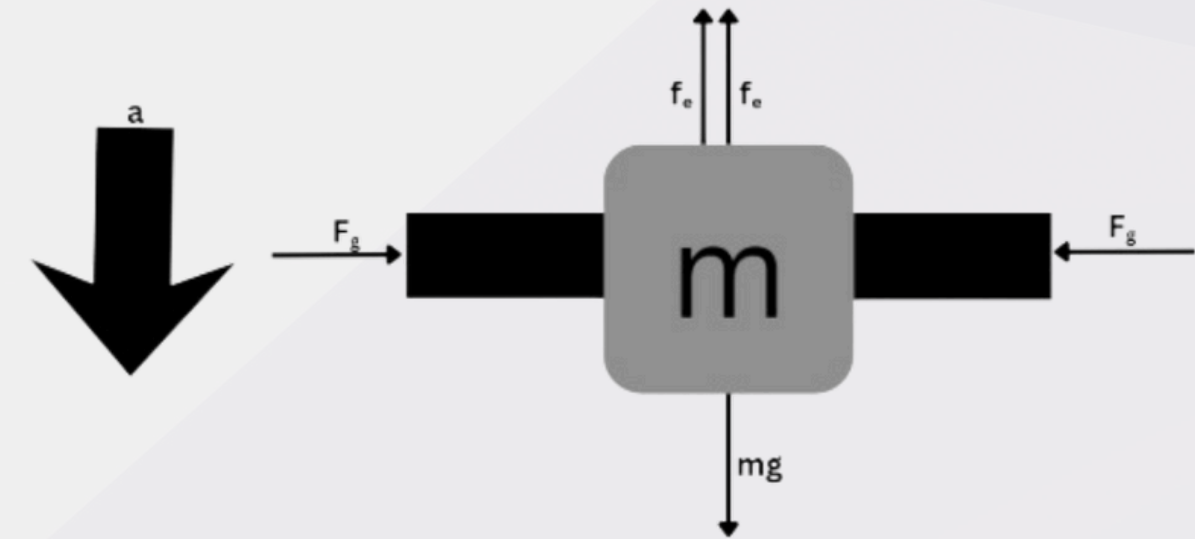
- Radio: $20 \text{ [cm]} = 0,2 \text{ [m]}$
- Masa: $30 \text{ [g]} = 0,03 \text{ [kg]}$
- Velocidad deseada:
 - $v = 0,3125 \text{ [m/s]}$
- Aceleración deseada:
 - $a = 1,5625 \text{ [m/s}^2\text{]}$
- Fuerza: $F = ma = 4,69 * 10^{-2} \text{ [N]}$
- Torque: $T = Fr = 9,38 * 10^{-3} \text{ [N]}$

FUERZA DE GARRA

- Coeficiente de fricción: $\mu = 0,6$

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

$$f_e < \mu N \longrightarrow f_e < F_g \mu \longrightarrow m < 2F_g \mu / ga$$



$$m_{\max} = 0,39 F_{gT}$$
$$m_{\max} = 0,78 \text{ [kg]}$$

DISEÑO DE LA INTERFAZ GRAFICA (BOCETO)



Boceto 1



Boceto 2



Boceto 3



Boceto 4

DISEÑO DE LA INTERFAZ GRAFICA (IMPLEMENTACIÓN)



DISEÑO DE LA INTERFAZ GRAFICA (BOCETO Y IMPLEMENTACION)



CLIENTE

```
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
```

CLIENTE

```
async def traducir_teclas():
    hub = await conectar_hub()
    if not hub:
        return

    while True:
        tecla = input("Comando: ").strip().lower()

        if tecla == "p":
            break
        elif tecla == "a": await enviar_programa(hub, "base", "positivo")
        elif tecla == "d": await enviar_programa(hub, "base", "negativo")
        elif tecla == "w": await enviar_programa(hub, "brazo", "positivo")
        elif tecla == "s": await enviar_programa(hub, "brazo", "negativo")
        elif tecla == "u": await enviar_programa(hub, "codo", "positivo")
        elif tecla == "j": await enviar_programa(hub, "codo", "negativo")
        elif tecla == "t": await enviar_programa(hub, "garra", "positivo")
        elif tecla == "g": await enviar_programa(hub, "garra", "negativo")
        else:
            print("Comando inválido.")

    await hub.disconnect()
```


CLIENTE

```
async def enviar_programa(hub, motor, direccion):
    programa_hub = generar_programa_hub(motor, direccion)

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

        await hub.run(ruta)

    finally:
        if 'ruta' in locals() and os.path.exists(ruta):
            os.remove(ruta)
```

SERVIDOR

```
def generar_programa_hub(motor: str, direccion: str) -> str:
    velocidades = {'base': 80, 'brazo': 60, 'codo': 70, 'garra': 50}
    puertos = {'base': 'A', 'brazo': 'D', 'codo': 'E', 'garra': 'C'}

    velocidad = velocidades.get(motor, 0)
    puerto = puertos.get(motor, "A")

    if direccion == "positivo":
        comando = f"motor.run({velocidad})"
    elif direccion == "negativo":
        comando = f"motor.run(-{velocidad})"
    else:
        comando = "motor.stop()"

    programa_hub= f"""
from pybricks.hubs import PrimeHub
from pybricks.pupdevices import Motor
from pybricks.parameters import Port
from pybricks.tools import wait

hub = PrimeHub()
motor = Motor(Port.{puerto})

{comando}
wait(300)
motor.stop()
"""

    return programa_hub
```


ESTADO ACTUAL DEL PROYECTO



Análisis y diseño

Definición de los objetivos, bibliotecas y realización de la garra robótica, con sus requerimientos necesarios.



Optimización

Pruebas del funcionamiento de la estructura, revisión de los movimientos del robot y actualizaciones de la carta Gantt.



Integración de software

Migración de código de bloques a Python para mejorar la flexibilidad y control sobre el robot, e integración del diseño de la interfaz gráfica.



Control externo

Se estableció la conexión por Bluetooth y se está mejorando la fluidez de los movimientos, además de la integración del controlador por mando.

PROBLEMAS ENCONTRADOS Y SOLUCIONADOS



Programación y control

Problemas en los movimientos y agarre del robot, en consecuencia de la coordinación de los motores.



Problema estructural

Fallos en la estructura del robot que afectaban la estabilidad general de su funcionamiento.



Entorno de desarrollo

Encuentro de limitaciones del entorno de LEGO y dificultad al programar los movimientos del robot mediante teclado.



Control externo

Dificultades al integrar la interfaz gráfica y problemas al intentar limitar las acciones del robot.

CONCLUSIONES

**MUCHAS
GRACIAS**

