

Carrito Inteligente Controlado con Micro Python y Raspberry Pi Pico

Jonathan Esteban Mojica Avendaño, código: 20222005109

Jeison Stiven Gómez Silva, código: 20231005080

Santiago Marulanda Parra, código: 20212005047

Proyecto Final de Programación Aplicada

Alfonso Peña Suarez

Universidad Distrital Francisco José De Caldas

Introducción

Este proyecto tiene como objetivo el diseño y la implementación de un vehículo inteligente que pueda ser controlado de manera remota mediante un teléfono móvil, utilizando una conexión Bluetooth o WiFi. La base del sistema es la Raspberry Pi Pico, un microcontrolador versátil y eficiente que permitirá integrar funcionalidades avanzadas de control y comunicación.

El carrito contará con capacidades básicas de movimiento, como avanzar, retroceder, girar a la derecha e izquierda, así como detenerse. Además, se desarrollará una interfaz personalizada para el control remoto a través de una aplicación móvil, lo que ofrecerá al usuario una experiencia interactiva y práctica.

Programado en MicroPython, este proyecto busca combinar tecnología, electrónica y programación para ofrecer un vehículo controlado de forma remota, capaz de ejecutar tanto comandos en tiempo real como patrones de movimiento preprogramados.

Problema

En muchas actividades cotidianas, como el transporte de pequeños objetos o la automatización de tareas simples en el hogar o la oficina, las soluciones tradicionales requieren supervisión manual o equipos costosos. No existen opciones accesibles y personalizables que permitan realizar estas tareas de forma remota y automatizada de manera sencilla.

Descripción del Problema

El transporte de objetos pequeños en distancias cortas o dentro de espacios reducidos, como casas, oficinas o talleres, a menudo requiere intervención manual, lo que consume tiempo y esfuerzos innecesarios. Aunque existen soluciones tecnológicas avanzadas, como robots

comerciales, estas suelen ser costosas, complejas y poco adaptables a las necesidades específicas del usuario.

Este proyecto busca solucionar esta problemática mediante la creación de un carrito controlado remotamente con un teléfono móvil. Este vehículo puede ser utilizado para transportar objetos ligeros dentro de un espacio definido, utilizando una conexión Bluetooth o WiFi. Al ser programable y personalizable, ofrece una solución práctica, accesible y adaptable a diversas tareas del día a día, facilitando así la automatización de actividades cotidianas con un enfoque sencillo y asequible.

Objetivo General

Desarrollar un carrito controlado de forma remota mediante conexión Bluetooth o WiFi, utilizando una Raspberry Pi Pico, como una solución práctica y accesible para automatizar el transporte de pequeños objetos en espacios reducidos, simplificando tareas cotidianas en el hogar, oficina o taller.

Objetivos Específicos

1. Diseño y construcción del carrito funcional:

- Diseñar una estructura ligera pero resistente para el carrito, capaz de transportar pequeños objetos de forma eficiente en distancias cortas.
- Integrar los componentes mecánicos y electrónicos, como motores, ruedas y la Raspberry Pi Pico, garantizando la funcionalidad del sistema de movimiento.

2. Desarrollo del sistema de control remoto personalizado:

- Establecer una conexión inalámbrica mediante Bluetooth o WiFi entre el carrito y un teléfono móvil, permitiendo un control remoto en tiempo real.
- Crear una interfaz intuitiva en una aplicación móvil que facilite al usuario manejar el carrito, con comandos básicos como avanzar, retroceder, girar y detenerse.

3. Programación y optimización del movimiento:

- Implementar un sistema de control programado en MicroPython que garantice precisión en los movimientos del carrito.

4. Pruebas y mejora de la solución:

- Evaluar el desempeño del carrito en entornos cotidianos (hogares, oficinas o talleres), verificando su capacidad para transportar objetos de manera eficiente y segura.
- Identificar y solucionar posibles problemas en la conectividad, estabilidad o precisión del movimiento, realizando ajustes para mejorar la experiencia del usuario.

5. Promoción de la accesibilidad y utilidad del proyecto:

- Documentar el diseño y funcionamiento del carrito, proporcionando una guía práctica para que otras personas puedan replicarlo o adaptarlo a sus necesidades.
- Enfocar el proyecto en resolver problemas reales, como la automatización de tareas sencillas, promoviendo soluciones tecnológicas asequibles y fáciles de implementar.

Materiales:

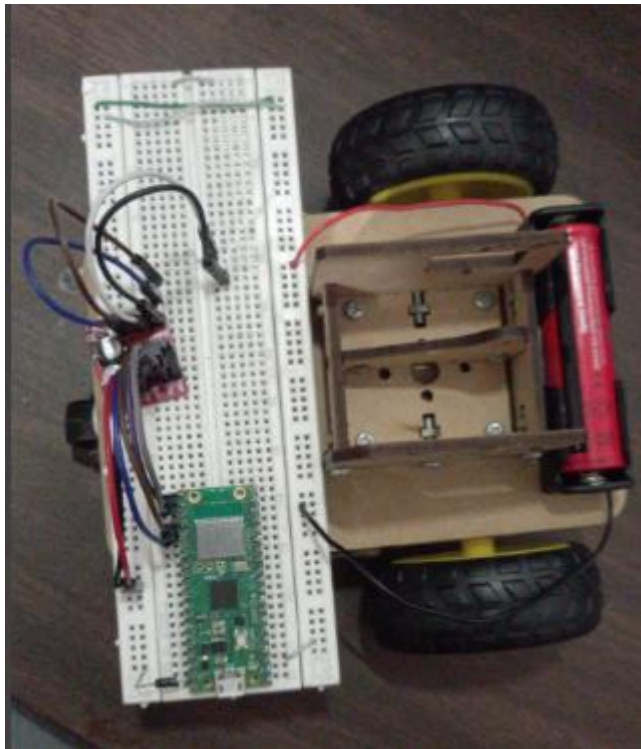
Hardware:

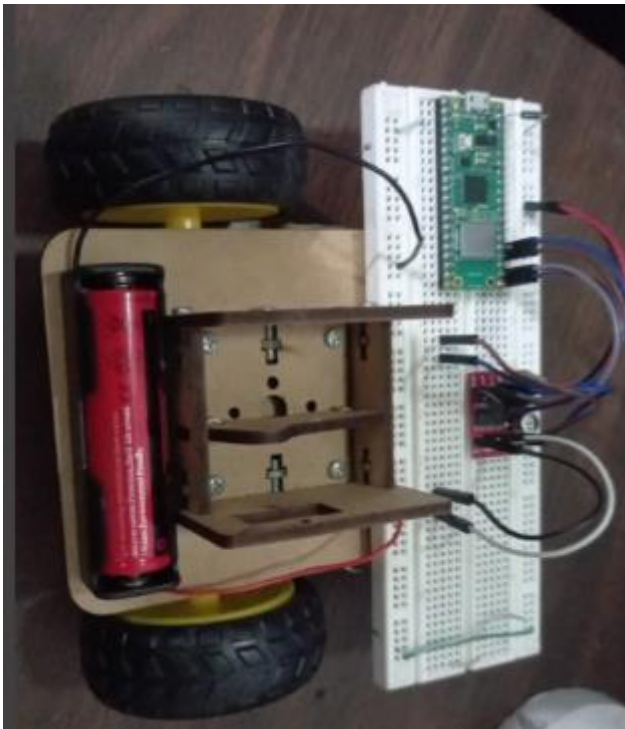
- Raspberry Pi Pico
- 2 Motores DC con ruedas
- 1 Rueda loca
- L298 (puente H) para controlar los motores
- Batería recargable de 7.5V
- Protoboard
- Jumper
- Madera

Software:

- MicroPython
- HTML

Diseño del producto





Página Web para controlar el carro (HTML):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Página con Tres Columnas</title>
  <style>
    body {
      display: grid;
      grid-template-columns: 1fr 1fr 1fr;
```

```

        grid-gap: 10px;
        margin: 20px;
    }

    .columna {
        padding: 10px;
        border: 1px solid #ccc;
    }

    button {
        width: 100%;
        padding: 10px;
    }
</style>
</head>
<body>

    <div class="columna">
        <input id="entrada1" type="text" value="1000" maxlength="6" >
    </div>
    <div class="columna">
        <!-- Segundo Renglón -->
        <button onclick="move_motors(1,1)">&uarr;</button>
    </div>
    <div class="columna">
        <input id="entrada2" type="text" value="1000" maxlength="6">
    </div>
    <div class="columna">
        <!-- Tercer Renglón -->
        <button onclick="move_motors(1,0)">&larr;</button>
    </div>
    <div class="columna">
        <button onclick="move_motors(0,0)">STOP</button>
    </div>
    <div class="columna">
        <button onclick="move_motors(0,1)">&rarr;</button>
    </div>
    <div class="columna"></div>
    <div class="columna">
        <!-- Cuarto Renglón -->
        <button onclick="move_motors(-1,-1)">&darr;</button>
    </div>
    <div class="columna"></div>

    <div style="grid-column: span 3;">
        <input type="text" id="commands" value="+600+10000+10000_+600-10000-
10000_+200+3000-3000_+600+10000+10000_+600-10000-10000_+200+3000-
3000_+600+10000+10000_+600-10000-10000_+200+3000-3000_" size="80"/>
        <button onclick="draw()">Draw</button>
    </div>

```

```

    <div style="grid-column: span 3;">
    <input type="text" id="commands" value="+600+10000+10000_+200+3000-
3000_+600+10000+10000_+200+3000-3000_+600+10000+10000_+200+3000-3000_"
size="80"/>
    <button onclick="draw()">Draw</button>
    </div>
    <div style="grid-column: span 3;">
    <input type="text" id="commands" value="+500+10000-0000_+500+000-10000_"
size="80"/>
    <button onclick="draw()">Draw</button>
    </div>
    <script>
    function move_motors(x, y) {
    const valorEntrada1 = parseFloat(document.getElementById('entrada1').value);
    const valorEntrada2 = parseFloat(document.getElementById('entrada2').value);

    if (isNaN(valorEntrada1) || isNaN(valorEntrada2)) {
        console.error('Valores de entrada no son
numéricos',document.getElementById('entrada1').value,document.getElementById('entrada2'
).value);
        return;
    }

    console.log('Valores de entrada:', valorEntrada1, valorEntrada2);

    // Use fetch to send coordinates to the server
    fetch(`?x=${valorEntrada1 * x}&y=${valorEntrada2 * y}`)
        .then(response => {
            if (!response.ok) {
                throw new Error('Error sending coordinates to the server');
            }
            console.log('Coordinates sent successfully');
        })
        .catch(error => console.error(error));
    }

    function draw() {
        const commands = document.getElementById('commands').value;

        console.log('Valores de comando:', commands);

        // Use fetch to send coordinates to the server
        fetch(`?command=${commands}`)
            .then(response => {
                if (!response.ok) {
                    throw new Error('Error sending coordinates to the server');
                }
                console.log('Commands sent successfully');
            })

```



```

        .catch(error => console.error(error));
    }
</script>
</body>
</html>

```

Código para controlar las funcion la Raspberry Pi Pico (Controlar los Motores, Conexión Wi-Fi, entre otras cosas):

```

import rp2
import network
import ubinascii
import machine
import urequests as requests
import time
import socket
import ure
from machine import Timer
import math

tim = None

mot0dir0 = machine.PWM(machine.Pin(16, machine.Pin.OUT))
mot0dir1 = machine.PWM(machine.Pin(17, machine.Pin.OUT))
mot1dir0 = machine.PWM(machine.Pin(18, machine.Pin.OUT))
mot1dir1 = machine.PWM(machine.Pin(20, machine.Pin.OUT))
mot0dir0.freq(1000)
mot0dir1.freq(1000)
mot1dir0.freq(1000)
mot1dir1.freq(1000)
#k=1

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
# If you need to disable powersaving mode
# wlan.config(pm = 0xa11140)

# See the MAC address in the wireless chip OTP
mac = ubinascii.hexlify(network.WLAN().config('mac'),':').decode()
print('mac = ' + mac)

# Other things to query
# print(wlan.config('channel'))
# print(wlan.config('ssid'))
# print(wlan.config('txpower'))

```

```

# Load login data from different file for safety reasons
ssid = "Ejemplo"
pw = "12345678"

wlan.connect(ssid, pw)

# Wait for connection with 10 second timeout
timeout = 10
while timeout > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    timeout -= 1
    print('Waiting for connection...')
    time.sleep(1)

# Define blinking function for onboard LED to indicate error codes
def blink_onboard_led(num_blinks):
    led = machine.Pin('LED', machine.Pin.OUT)
    for i in range(num_blinks):
        led.on()
        time.sleep(.2)
        led.off()
        time.sleep(.2)

# Handle connection error
# Error meanings
# 0 Link Down
# 1 Link Join
# 2 Link NoIp
# 3 Link Up
# -1 Link Fail
# -2 Link NoNet
# -3 Link BadAuth

wlan_status = wlan.status()
blink_onboard_led(wlan_status)

if wlan_status != 3:
    raise RuntimeError('Wi-Fi connection failed ',wlan_status)
else:
    print('Connected')
    status = wlan.ifconfig()
    print('ip = ' + status[0])

# Function to load in html page
def get_html(html_name):
    with open(html_name, 'r') as file:
        html = file.read()

```

```
return html
```

```
def findall_1(pattern, text):
```

```
    matches = []
```

```
    start = 0
```

```
    while True:
```

```
        match = re.search(pattern, text[start:])
```

```
        if match is None:
```

```
            break
```

```
        matches.append(match.group(0))
```

```
        start += match.end()
```

```
    return matches
```

```
# HTTP server with socket
```

```
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
```

```
s = socket.socket()
```

```
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

```
s.bind(addr)
```

```
s.listen(1)
```

```
print('Listening on', addr)
```

```
led = machine.Pin('LED', machine.Pin.OUT)
```

```
def next_command(timer):
```

```
    global angle_LOGO_deg, P0, P1, t, comandos
```

```
    #commands=comandos
```

```
    print('comandos',comandos)
```

```
    if comandos != [] :
```

```
        tiem,m1,m2=comandos.pop(0)
```

```
        set_PWMs(mot0dir1,mot0dir0,m1)
```

```
        set_PWMs(mot1dir0,mot1dir1,m2)
```

```
        tim = Timer(period=abs(tiem), mode=Timer.ONE_SHOT, callback=next_command)
```

```
    else:
```

```

set_PWMs(mot0dir1,mot0dir0,0)
set_PWMs(mot1dir0,mot1dir1,0)
tim.deinit()
tim=None

```

```

def set_PWMs(mot_dir0,mot_dir1,x_value):
    print(mot_dir0,mot_dir1,x_value)
    if x_value<0:
        mot_dir0.duty_u16(-x_value)
        mot_dir1.duty_u16(0)
    elif x_value>0:
        mot_dir0.duty_u16(0)
        mot_dir1.duty_u16(x_value)
    else:
        mot_dir0.duty_u16(0)
        mot_dir1.duty_u16(0)

```

```

comandos=[]
# Listen for connections
while True:
    try:
        cl, addr = s.accept()
        led.on()

        print('Client connected from', addr)
        r = cl.recv(1024)

        r = str(r)
        print(r[:30])
        # Utilizar expresiones regulares para encontrar los valores de x e y
        match = ure.search(r'\?x=(-?\d+)&y=(-?\d+)', r)

        if match:
            x_value = int(match.group(1))
            y_value = int(match.group(2))

            print(f'Valor de x: {x_value}')
            print(f'Valor de y: {y_value}')

```

```

set_PWMs(mot0dir1,mot0dir0,x_value)
set_PWMs(mot1dir0,mot1dir1,y_value)

if tim:
    tim.deinit()
    tim=None

else:
    print('No se encontraron valores de x e y en la cadena de la solicitud HTTP.')

match = ure.search(r'command=([^\s]+)', r)
if match:
    input_string= match.group(1)
    matches = []
    start = 0

    while True:
        match = ure.search(r'([+-]?\d+)_?', input_string[start:])
        if match is None:
            break

        matches.append(int(match.group(1)))
        start += match.end()

    # Build the list of lists
    result = [matches[i:i+3] for i in range(0, len(matches), 3)]

    comandos += result#ure.finditer(r'([+-]?\d+)([+-]?\d+)([+-]?\d+)_?', match.group(1))
    #match.group(1).split('_')
    # triplas = re.findall(r'([+-]?\d+)([+-]?\d+)([+-]?\d+)_?', cadena)
    if not tim:
        next_command(None)
        #tim=Timer(period=5000, mode=Timer.ONE_SHOT, callback=lambda t:print(1))

else:
    print('No se encontraron comandos en la cadena de la solicitud HTTP.')

response = get_html('flecha.html')
cl.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n')
cl.send(response)
cl.close()
#print(r[:20])
#time.sleep(.2)
led.off()
#time.sleep(.2)
except OSError as e:
    cl.close()
    print('Connection closed')

```

```
# Make GET request
#request = requests.get('http://www.google.com')
#print(request.content)
#request.close()
```

Proceso de Construcción:

Se cortó un pedazo de madera de 18 cm de largo y 10 cm de ancho mediante láser para que sirviera de base del carro. También se fabricó otra pieza que funcionara como la 'caja' del carro, incluyendo los espacios necesarios para unir ambas partes con tornillos. En la parte inferior, se fijaron dos motoredutores junto con sus llantas, además de una rueda loca para estabilizar el carro. La batería se colocó en la parte trasera de la 'caja', mientras que la protoboard se ubicó en la parte delantera.

Conclusiones:

El proyecto se encuentra en una etapa inicial con avances significativos en la implementación técnica. El uso de la Raspberry Pi Pico y MicroPython ha demostrado ser adecuado para controlar un vehículo remoto, estableciendo una base funcional para el movimiento mediante PWM y la comunicación WiFi. Hasta ahora, se ha logrado ejecutar comandos en tiempo real lo que confirma la viabilidad del sistema.

Entre los desafíos identificados destacan la necesidad de optimizar la conectividad WiFi y mejorar la estabilidad del movimiento. Asimismo, es crucial integrar el diseño físico del carrito con los componentes electrónicos, realizar pruebas en entornos reales y ajustar el sistema según los resultados.

El proyecto ofrece una solución accesible y personalizable para la automatización de tareas sencillas, con potencial para ser replicado y adaptado. Con los próximos pasos enfocados en pruebas, refinamiento y documentación, se espera cumplir con el objetivo de

proporcionar una herramienta práctica y funcional para el transporte remoto de objetos pequeños en espacios reducidos.