



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA DEPARTAMENTO DE INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE MICROCOMPUTADORAS

Manual de recursos y aplicaciones PLATAFORMA RASPBERRY Pi

RUBÉN ANAYA GARCÍA MOISES MELENDEZ REYES ISRAEL RIVERA ALBERTO TEMPLOS CARBAJAL

> CIUDAD UNIVERSITARIA PE108223

Material generado en colaboración con los docentes de la academia de la asignatura de Microcomputadoras, integrado por profesores de la Teoría y Laboratorio; así como de profesores participantes.

RUBÉN ANAYA GARCÍA MOISES MELENDEZ REYES ANTONIO SALVA CALLEJA JOSE ANTONIO ARREDONDO GARZA ANGELICA QUIÑONES JUAREZ DIANA CRUZ HERNANDEZ LUIS SERGIO DURÁN ARENAS ROMÁN OSORIO COMPARAN ALBERTO TEMPLOS CARBAJAL

Este material y la infraestructura generada fueron obtenidos a través de apoyo de DGAPA al proyecto PAPIME PE108223 "Prácticas de Laboratorio de microcomputadoras basadas en las plataformas Raspberry Pi"

Introducción

Este manual proporciona información útil en la implementación del proyecto PE108223: "Practicas de Laboratorio de Microcomputadoras: Basadas en las Plataformas Raspberry Pi", en el que se presentan los procedimientos e información complementaria para la realización de las actividades solicitadas.

Contenido:
1. Plataformas Raspberry Pi.
2. Descarga, instalación de Raspberry SO.
3. Conexión remota.
4. Ensamblador y GDB.
5. IDE Code::Blocks.
6. Simulador en línea CPUlator.
7. Descarga e instalación de Thonny.
8. Programación de APP inventor; comunicación Bluetooth.
9. Control remoto de giro de un motor a pasos.
10. Control manual de posición angular de un motor a pasos.
11. Giroscopio y acelerómetro.
12. Control de un motor de CD.

1

Raspberry Pi

Plataformas

La plataforma Raspberry ha sido diseñada por la Fundación Raspberry; con el paso del tiempo ha diseñado distintas versiones:

1. Raspberry Pi

Plataforma	SoC	Memoria	E/S	Conectividad
Raspberry Pi modelo B, B+	BCM2835	256 MB 512 MB	26	HDMI, 2 USB 2.0, audio, video RCA, Ethernet RJ45, SD Card, Alimentación micro USB
Raspberry Pi modelo A, A+	BCM2835	256 MB	26	HDMI, 2 USB 2.0, audio, video RCA, SD Card, Alimentación micro USB
Raspberry Pi 2 Modelo B	BCM2836, BCM2837	1 GB	40	HDMI, 4 USB 2.0, audio, video RCA, Ethernet RJ45, SD Card, Alimentación micro USB
Raspberry Pi 3 modelo B, B+	BCM2837 BCM2837b0	1 GB	40	HDMI, 2 USB 2.0, audio, video RCA, Ethernet RJ45, WiFi 2.4 GHz, Bluetooth 4.1, Bluetooth BLE, SD Card, Alimentación micro USB
Raspberry Pi 4 modelo B	BCM 2711	1GB, 2GB, 4GB y 8GB	40	2 micro HDMI, 2 USB 2.0, 2 USB 3.0, audio, video RCA, Ethernet RJ45, WiFi 2.4/5 GHZ (120 Mb/s) SD Card, Alimentación micro USB
Raspberry Pi 4	BCM 2712	2GB, 4 GB, 8 GB Y 16 GB	40	2 micro HDMI, 2 USB 2.0, 2 USB 3.0, audio, video RCA, Ethernet RJ45, WiFi 2.4/5 GHZ (300 Mb/s) SD Card, Alimentación tipo C

Tabla 1. Características generales de las versiones microcomputadoras

Manual de Aplicaciones Raspberry Pi



Raspberry Pi 5



Raspberry Pi 4





Raspberry Pi 1



2. Versiones Raspberry + Teclado

Plataforma	SoC	Memoria	E/S	Conectividad
Raspberry Pi 400	BCM 2711	4GB	40	2 micro HDMI, 2 USB 2.0, 2 USB 3.0, audio, video RCA, Ethernet RJ45, WiFi 2.4/5 GHZ (120 Mb/s) SD Card, Alimentación micro USB
Raspberry Pi 500	BCM 2712	8GB	40	2 micro HDMI, 2 USB 2.0, 2 USB 3.0, audio, video RCA, Ethernet RJ45, WiFi 2.4/5 GHZ (300 Mb/s) SD Card, Alimentación tipo C

Tabla 2. Versiones integradas



Raspberry Pi 500.

Raspberry Pi 400.

Figura 2. Imágenes de Raspberry integradas

3. Versiones reducidas (Zero).

Plataforma	SoC	Memoria	E/S	Conectividad
Raspberry Pi Zero	BCM2835	512 MB	40	Puerto mini HDMI, 2 puertos micro USB, slot tarjetas micro SD
Raspberry Pi Zero W	BCM2835	512 MB	40	Puerto mini HDMI, 2 puertos micro USB, slot tarjetas micro SD, WiFi 2.4 GHz (35 mb/s), Bluetooth 4.0, Bluetooth BLE
Raspberry Pi Zero 2W	RP3A0	512 MB	40	Puerto mini HDMI, 2 puertos micro USB, slot tarjetas micro SD, WiFi 2.4 GHz (35 mb/s), Bluetooth 4.2, Bluetooth BLE

Tabla 3. Características Raspberry Pi Zero



Raspberry Pi Zero



Raspberry Pi Zero W



Raspberry Pi Zero 2W

Figura 3. Imágenes Raspberry Pi Zero

4. Versiones microcontrolador

Plataforma	SoC	Memoria	ROM	E/S	Cone	ctivida	d		
Raspberry Pi Pico	RP2040	264 MB	2 MB	40/26	-				
Raspberry Pi Pico W	RP2040	264 MB	2 MB	40/26	WiFi	2.4	GHz	(10	Mb/s),
					Blueto	oth 5.	2, Blue	tooth	BLE
Raspberry Pi Pico 2	RP2350	520 MB	4 MB	40/26	-				
Raspberry Pi Pico 2 W	RP2350	520 MB	4 MB	40/26	WiFi	2.4	GHz	(10	Mb/s),
					Blueto	oth 5.	2, Blue	tooth	BLE

Tabla 4. Características Raspberry Pi Pico





Raspberry Pi Pico 2

Raspberry Pi Pico 2W

Figura 4. Imágenes Raspberry Pï Pico

2

Sistema Operativo Raspberry Pi Raspberry Pi OS

La instalación requiere de:

- 1. Descarga del SO.
- 2. Instalación de imagen.
- 3. Instalación del SO en la unidad micro SD.

1.Descarga del SO.

La imagen del sistema operativo se puede descargar del sitio oficial de Raspberry Pi, existe versiones para Windows, MacOS y Ubuntu:

https://www.raspberrypi.com/software/

Una vez seleccionado el sistema operativo de su computadora, iniciará el proceso de descarga.



Una vez concluido el proceso de descarga, tendrá a disposición el archivo ejecutable.



Figura 2. Imagen del SO

2. Instalación Raspberry Pi Imager.

Al Ejecutar el programa .exe, será confirmada la acción; presionar **Install** y esperar a concluir la instalación.



Figura 3. Instalación de la imagen



Figura 4. Instalación Raspberry Pi Imager, concluido

3. Instalación de Raspberry Pi OS.



Figura 5. Entorno del instalador

El proceso de instalación consta de las siguientes etapas; estas se muestran en las figuras 6 a la 17.

- a) Selección de la versión de Raspberry.
- b) Selección de la versión del SO.
- c) Selección de la unidad donde se ubica la memoria micro SD.
- d) Configuraciones durante el proceso de instalación del SO, entre los que se encuentra:
- e) Credenciales de identificación (nombre y contraseña).
- f) Red de conexión alámbrica o inalámbrica.
- g) Habilitación de comunicación SSH.
- h) Confirmación de configuraciones.

i) Instalar en Raspberry Pi.







Figura 7. Selección del Sistema Operativo



Figura 8. Selección de la unidad micro SD



Figura 9. Configuraciones generales

Raspberry Pi Imager v	Raspberry Pi			×
Dispo	¿Usar la personalización del SO? ¿Desea aplicar los ajustes de personalización del SO? DITAR AJUSTES NO, BORRAR AJUSTES SÍ NO	x	£	
	SIGUIENTI			

Figura 10. Confirmación de personalización

Seleccionar editar ajustes para configurar claves de acceso a Raspberry, conexión inalámbrica, activar SSH y algunas configuraciones adicionales (figura 11).

ersonalización del SO		÷ 0	Personalización del SO		- 0
GENERAL	SERVICIOS	OPCIONES	GENERAL	SERVICIOS	OPCIONES
Establecer nombre	de anfitrión: raspberrypi	.local	Activar SSH		
Establecer nombre	de usuario y contraseña		🔘 Usar autenti	cación por contraseña	
Nombre de usuario: P	i		O Permitir sol	o la autenticación de clave p	ública
Contraseña:	•		Establecer aut	horized_keys para 'pi':	
🗹 Configurar LAN inal	ámbrica				
SSID:	micros		EJECUTA	R SSH-KEYGEN	
Contraseña:	•••••				
Mostrar cont	raseña 🔲 SSID oculta				
País de LAN inalámbrica	· MX ·				
Establecer ajustes r	regionales				

nalización del SO		÷ 11
GENERAL	SERVICIOS	OPCIONES
Reproducir sonido a	ıl finalizar	
Expulsar soporte al	finalizar	
🗹 Activar telemetría		
	CONTROL OF	

Figura 11. Personalizaciones iniciales en el SO

Cuando termine las configuraciones; se solicitará la confirmación. ${\scriptstyle[\![]]{3.5ex}}$ Raspberry Pi Imager v18.5



Figura 12. Ventana de confirmación de personalizaciones

Será iniciado la instalación.

	Advertencia	x	
RA	Se borrarán todos los datos existentes en 'Generic- SD/MMC USB Device'. ¿Está seguro de que desea continuar?		E

Figura 13. Confirmación de todo el proceso

Una ver configurado y confirmado, el sistema operativo será instalado en la unidad micro SD.



Figura 14. Ventana de preparación

Manual de Aplicaciones Raspberry Pi

Ras	oberry Pi	
Dispositivo Raspberry Pi RASPBERRY Pi 4	Sistema operativo RASPBERRY PI OS (32-BIT)	Almacenamiento

Figura 15. Ventana del estado de instalación

Raspberry Pi Imager v1.8.5	berry Pi	~ 0	×
Dispositivo Raspberry Pi RASPBERRY PI 4	Sistema operativo RASPBERRY PI OS (32-BIT)	Almacenamiento GENERIC- SD/MMC USB DEVICE	
Preparando para esc	ribir (Personalizando imagen)	CANCELAR VERIFICACIÓN	

Figura 16. Proceso terminado

Manual de Aplicaciones Raspberry Pi



Figura 17. Instalación completa

Como último paso en este proceso, será la extracción de la unidad micro SD de la computadora y conectarla a la Raspberry para concluir con la instalación en esa plataforma..

3

Raspberry Pi

Acceso Remoto

La conexión remota de Raspberry Pi usando un equipo de cómputo externo facilita en gran medida el uso de la plataforma, es un mecanismo de control cuando no se dispone de teclado, mouse o pantalla.

Entre los servicios de acceso remoto más empleados se encuentran:

- I. SSH (Secure Shell); permite acceso a la terminal de Raspberry Pi.
- II. VNC (Virtual Network Computing); acceso al ambiente grafico de Raspberry Pi.

En ambos casos, es necesario conectarse a la red y conocer la dirección IP asignada a la tarjeta Raspberry Pi; existen varios métodos para identificarlo, podrá elegir el de su preferencia.

Contenido:

- 1. Habilitar SSH.
- 2. Conexión SSH.
- 3. Habilitar VNC.
- 4. Descargar y habilitar VNC Server.
- 5. Conexión VNC Server.

1. Activación SSH.

Se requiere habilitar en la Raspberry Pi el protocolo SSH; métodos:

I. Al momento de configurar la instalación del sistema operativo.

Establecer nombre de anfitrión: raspberrypi .local Establecer nombre de usuario y contraseña Nombre de usuario: PÍ Nombre de usuario: PÍ		GENERAL	SERVICIOS	OPCIONE
Establecer nombre de usuario y contraseña Nombre de usuario: Director Contraseña: SSID: micros Contraseña: Ontraseña: Mostrar contraseña SSID oculta País de LAN inalámbrica: MX País de LAN inalámbrica: MX Mastrar contraseña SSID oculta País de LAN inalámbrica: MX Contraseña: MIX País de LAN inalámbrica: MX Contraseña: Decionales		Establecer nombr	e de anfitrión: raspberrypi	.local
Nombre de usuario: pi Contraseña: •• Configurar LAN inalámbrica SSID: micros Contraseña: ••••••••••••• Mostrar contraseña SSID oculta País de LAN inalámbrica: MX País de LAN inalámbrica: MX Establecer ajustes regionales	~	Establecer nombr	e de usuario y contraseña	
Contraseña: Configurar LAN inalámbrica SSID: Contraseña: Contraseña: Mostrar contraseña SSID oculta País de LAN inalámbrica: MX Establecer ajustes regionales		Nombre de usuario:	pi	
Configurar LAN inalámbrica SSID: miCros Contraseña: Mostrar contraseña SSID oculta País de LAN inalámbrica: MX Establecer ajustes regionales ización del SO GENERAL SERVICIOS OPCIONES		Contraseña:	••	
SSID: micros Contraseña: SSID oculta Mostrar contraseña SSID oculta País de LAN inalámbrica: MX • Establecer ajustes regionales kración del SO - GENERAL SERVICIOS OPCIONES	~	Configurar LAN in	alámbrica	
Contraseña: ••••••••••••••••••••••••••••••••••••		SSID:	micros	
☐ Mostrar contraseña ☐ SSID oculta País de LAN inalámbrica: MX ✓ ✓ Establecer ajustes regionales ✓ tración del SO – GENERAL SERVICIOS OPCIONES				
País de LAN inalámbrica: MX Establecer ajustes regionales Ización del 50 GENERAL SERVICIOS OPCIONES		Contraseña:		
País de LAN inalámbrica: MX Establecer ajustes regionales Iización del SO GENERAL SERVICIOS OPCIONES		Contraseña:	ntraseña 🔲 SSID oculta	
GENERAL SERVICIOS OPCIONES	V	Contraseña: Mostrar con País de LAN inalámbri Establecer ajustes	ntraseña SSID oculta ica: MX – s regionales	
	Ización	Contraseña: Mostrar con País de LAN inalámbri Establecer ajustes	ntraseña 🗍 SSID oculta ica: MX 👻	
	lización A	Contraseña: Mostrar con País de LAN inalámbri Establecer ajustes Idol 50 GENERAL	ntraseña 📄 SSID oculta ica: MX 👻 s regionales SERVICIOS	OPCIONES
 Usar autenticación por contraseña 	Ización	Contraseña: Mostrar con País de LAN inalámbri Establecer ajustes idel 50 GENERAL Activar SSH O Usar autentic	eternological de la construcción por contraseña	OPCIONES
 Usar autenticación por contraseña Permitir solo la autenticación de clave pública 	Ilización	Contraseña: Mostrar con País de LAN inalámbri Establecer ajustes e del SO GENERAL 	Antraseña SSID oculta ica: MX	OPCIONES
 Usar autenticación por contraseña Permitir solo la autenticación de clave pública Establecer authorized_keys para 'p': 	Ilización	Contraseña: Mostrar con País de LAN inalámbri Establecer ajustes Idel 50 GENERAL Citivar SSH Usar autentic Permitir solo Establecer autho	Antraseña SSID oculta ica: MX	- OPCIONES
 Usar autenticación por contraseña Permitir solo la autenticación de clave pública Establecer authorized_keys para 'pl': 	Rización	Contraseña: Mostrar con País de LAN inalámbri Establecer ajustes idel 50 GENERAL Citivar SSH Usar autentic Permitir solo Establecer auto	Antraseña SSID oculta ica: MX	OPCIONES
 Usar autenticación por contraseña Permitir solo la autenticación de clave pública Establecer authorized_keys para 'pi': EJECUTAR SSH-KEYGEN 	Ización	Contraseña: Mostrar con País de LAN inalámbri Establecer ajustes del SO GENERAL CUSar autentic Permitir solo Establecer auto Establecer auto	Ación por contraseña la autenticación de clave públi prized_keys para 'pi':	OPCIONES

Figura 1. Activar SSH

II. Una vez arrancado el SO, seleccionar configuration dentro de preferences.

Manual de Aplicaciones Raspberry Pi



Raspberry Pi Configuration 🔷 🔺 🗙							
System	Display	Interfaces	Performance	Localisation			
SSH:							
VNC:				\bigcirc			
SPI:				\bigcirc			
12C:				\bigcirc			
Serial Port:				\bigcirc			
Serial Cons	ole:			\bigcirc			
1-Wire:				\bigcirc			
Remote GP	10:						
			Cano	cel OK			

Figura 2. Configuración Raspberry Pi

2. Conexión SSH.

A. Conectar a través del SSH.

Abrir la terminal (cmd) en Windows, escribir ssh pi@ip_asignado a la Raspberry Pi.





Solicitará las credenciales de acceso, al ser ingresados correctamente, tendrá el control de la plataforma (figura 6).



Figura 4. Control remoto

3. Habilitar VNC.

a. Habilitar VNC; seguir el mismo procedimiento para SSH.

Raspberry Pi Configuration 🛛 🗸 🔺 🗙							
System	Display	Interfaces	Performance	Localisation			
SSH:							
VNC:							
SPI:				\bigcirc			
12C:				\bigcirc			
Serial Port:				\bigcirc			
Serial Cons	ole:			\bigcirc			
1-Wire:				\bigcirc			
Remote GP	10:			\bigcirc			
			Cano	el OK			



- b. Configurar VNC a través de la conexión SSH.
 - *i.* Usar procedimiento descrito en 1.1.
 - ii. Habilitar usando el menú de configuraciones.

Manual de Aplicaciones Raspberry Pi

Raspberry Pi Soft	ware Configuration Tool (raspi-config)	
1 System Options 2 Display Options 3 Interface Options 4 Performance Options 5 Localisation Option 6 Advanced Options 8 Update 9 About raspi-config	Configure system settings Configure display settings Configure connections to paripherals Configure performance settings s Configure language and regional settings Configure advanced settings Update this tool to the latest version Information about this configuration tool	
(Select)	(Finish)	

🖬 pi@raspberrypi: ~		7 2	×
II SSH II SSH II RPI Connect II SPI IS IZC IS SETAL PORT IS FICA IS Serial Port IS Remote GPIO	PI softwahe conregulation (oc) (respiceoring) p nable/disable remote command line access using SSH nable/disable Rasphery Pi Connect nable/disable graphicel remote desktop access nable/disable automatic loading of SJI kernel module nable/disable shell messages on the serial connection nable/disable one-wire interface nable/disable nemote access to GPIO pins		
<sel< td=""><td>t> <back></back></td><td></td><td></td></sel<>	t> <back></back>		



Figura 6. Habilitar server VNC

4. Descargar e instalar VNC Server.

Instalar VNC Server / Real VNC Viewer; disponible en línea; https://www.realvnc.com/en/connect/download/viewer/. Desktop Mobile A 8 Windows Raspberry Pi Linux macOS EXE x86/x64 🕹 Download RealVNC Viewer Figura 7. Descarga RealVNC Viewer

- Seleccionar el sistema operativo de su equipo de cómputo. i.
- ii. Esperar la descarga e instalar.

5. Conexión con el servidor VNC.

Ejecutar RealVNC Viewer para conectar a la Raspberry; introducir la dirección IP; se solicitarán las credenciales de acceso.

VNC CONNECT 192.168.1.176		🌡 (niciar sesión 🔹
BE Directiones Conect	arse a la dirección o el nombre de host *192.168.1.176*	
		the second
		92.158.1.147
	micros micros2	
	19679/2000 (19817)-1997-1997	
Necesitas avuda para conectartei	×	
form mailtin on feet		
dispositivo(s)		
dispositivo(s)	Figura 8. Ingreso IP.	
Real/VNC Viewer	Figura 8. Ingreso IP.	- a x
RealVINC Viewer hivo Visualizar Ayuda	Figura 8. Ingreso IP.	- I ×
Real/WC Viewer hivo Visualizar Ayuda http://www.commerce.com/ecci. D20 Newsciewer	Figura 8. Ingreso IP.	- I X
ResIVMC Viewer New Visualizar Ayuda Inc connect 192.168.1.176 B Directiones	Figura 8. Ingreso IP.	- C ×
Ispositive(s) Real/VHC Viewer have Visualizar Ayuda https://www.realizar Ayuda therefore the the the the the the the the the th	Figura 8. Ingreso IP.	- D ×
RealVNC Viewer Neo Visualizar Ayuda CCONNECT 192.168.1.176 Direcciones	Figura 8. Ingreso IP.	- C ×
dispositivo(s) Peal/VIC Viewer chieo Visualizar Ayuda mcconnect 192.168.1.176 Directiones	Figura 8. Ingreso IP.	×
dispositivo(s) Peal/WC Viewer chiero Visualizar Ayuda Inc connect 192.168.1.176 Directiones	Figura 8. Ingreso IP.	 Iniciar sesión. Iniciar sesión. Iniciar sesión.
dispositive(s) Real/VHC Viewer thivo Visualizar Ayuda InC CONNECT 192.168.1.176 Direcciones	Figura 8. Ingreso IP.	 Iniciar sesión. Iniciar sesión. Iniciar sesión.
dispositivo(s) Real/VHC Viewer chivo Visualizar Ayuda InC CONNECT 192.168.1.176 Direcciones	Figura 8. Ingreso IP.	 Iniciar sesión Iniciar sesión Iniciar sesión
Ilepainve(s) PeainVAC Viewer thivo Visualizar Ayuda CCONNECT 192.168.1.176 Direcciones	Figura 8. Ingreso IP.	 Iniciar sesión. Iniciar sesión. Iniciar sesión.
dispositivo(s) Pea/VVC Viewer chiro Visualizar Ayuda Inc connect 192.168.1.176 Direcciones	Figura 8. Ingreso IP.	×
dispositive(s) Peal/VHC Viewer thiro Visualizar Ayuda Inc connect 192.168.1.176 Directiones	Figura 8. Ingreso IP.	× ♥ Iniciar sesión. ▼ 192.168.1.147
RealVWC Viewer hivo Visualizer Ayuda Inc connect 192.168.1.176 BE Directiones	Figura 8. Ingreso IP.	× ▲ Iniciar sesión. ▼ 192.168.1.147
RealVWC Viewer hivo Visualizer Ayuda Inc connect 192.168.1.176 B Directiones	Figura 8. Ingreso IP.	× Iniciar sesión.
dispositivo(s) Real/VHC Viewer thivo Visualizar Ayuda INC CONNECT 192.168.1.176 Direcciones	Figura 8. Ingreso IP.	× Iniciar sesión ▼ 192.168.1.147
dispositive(s) Peal/WC Viewer thivo Visualizar Ayuda InC CONNECT 192.168.1.176 Directiones	Figura 8. Ingreso IP.	× ▲ Iniciar sesion. • 192.168.1.147
RealVNC Viewer hivo Visualizar Ayuda Inc connect 192168.1.176	Figura 8. Ingreso IP.	Initiar sesion.

En esta etapa, tendrá control remoto de la Raspberry Pi.

Manual de Aplicaciones Raspberry Pi



Figura 10. Escritorio del Sistema Operativo

4

Raspberry Pi

Ensamblador, Ligador y Debugger GDB

Editar y ensamblar un programa se puede realizar por varios mecanismos, a continuación, se estudiará el procedimiento que permitirá emplear los recursos del paquete GNU GCC Compiler y servicios del sistema operativo.

A. Editor de texto.

Existen una gran variedad de editores para captura, entre los sugeridos se encuentran:

1. Editor vi.

Es un editor que se ejecuta sobre una terminal.

- Se invoca desde la línea de comandos de la consola:
 - vi NombrePrograma.s



Figura 1. Editor vi

2. Editor nano.

Es un editor que se ejecuta sobre una terminal.

- Se invoca desde la línea de comandos de la consola.
 - nano NombrePrograma.s



Figura 2. Editor nano

3. Editor Geany.

Es un ambiente gráfico; se ubica en el software de programación del Sistema Operativo de Raspberry Pi.

									plantill	ag.s - /h	ome/pi - G	ieany	
File Edit Search	View	Document I	Project Buil	d Tools	Help								
<u> </u>	4		× <	\rightarrow	<u> </u>	• •			<u> </u>		4	3	•
	р	lantillag.s ×				e more l:	ons						
<pre></pre>	1 2 3 4 5 6 7	.text .global _start: .data	_start										
									Fig	ura 3	3. Edit	or G	eany

El procedimiento a seguir independientemente de editor elegido:

- 1. Editar el programa.
- 2. Ensamblar
- 3. Ligar
- 4. Ejecutar en línea o con el debugger gdb.

a. Edición de un programa.

Escribir el código empleando el editor de su preferencia, guardar con la extensión .s.

8		Practicas	🗾 pi@ra	spberrypi: ~/D	🧓 e1.s - /home/p	oi/Desk 🧯	3	()
			pi@raspbe	errypi: ~/Desktop/	Practicas		Ì	
File	Edit Tabs Help							
GNU	nano 7.2			e51.s				
	.text							
_start	:: mov r0,#5 mov r1,#0x01 subs r3,r0,r1 beq igual							
igual:	bne diferente mov r0,#1 ldr r1,=texto1 mov r2,#30 mov r7,#4 svc 0 b fin							
difere	ente: mov r0,#1 ldr r1,=texto2 mov r2,#33 mov r7,#4 svc 0							
fin:	mov r0,r3 mov r7,#1 svc 0 .data							
texto1 texto2	l: .ascii "Datos igu ?: .ascii "Datos dif	ales resultado erentes resulta	= " ado = "					
^G Hel ^X Exi	Lp <mark>^O</mark> Write On Lt <mark>^R</mark> Read Fi	ut <mark>AW</mark> Where Is le <mark>AN</mark> Replace	<mark>∧K</mark> Cut ∧U Paste	<mark>∧T</mark> Execute ∧J Justify	^C Location ☆ Go To Line	M-U Undo M-E Redo	M-A M-6	Set Mark Copy



b. Ensamblado.

Generar el código objeto, usando el ensamblador as.

as -o NombrePrograma.o NombrePrograma.s

pi@raspberrypi:~/Desktop/Practicas \$ as -o ejemplo.o ejemplo.s

Figura 5. Ensamblado

c. Ligado.

Generar el código ejecutable; se invoca el ligador Id.

Id -o NombrePrograma NombrePrograma.o

pi@raspberrypi:~/Desktop/Practicas \$ ld -o ejemplo ejemplo.o

Figura 6. Ligado

En caso de no existir errores el proceso de ensamblado y ligado, no será desplegado alguna información de salida; en caso contrario enviará los mensajes de error, para su atención.

d. Ejecución.

Con el código ejecutable NombrePrograma (no tendrá extensión), se ejecutará el programa y se comprobará el resultado obtenido, deberá teclear:

.WombrePrograma ; echo \$?

pi@raspberrypi:~/Desktop/Practicas \$./ejemplo ; echo \$?
Datos diferentes ... resultado = 4

Figura 7. Ejecución

Por lo tanto, el procedimiento completo se muestra en la figura 8:



```
Figura 8. Etapas
```

En caso que un programa no haga uso de los servicios, librerías y prestaciones del GNU GCC Compiler de manera directa, será más sencillo la ejecución por pasos del programa mediante el uso de GDB.

B. Ejemplo.

Para explicar el uso de GDB; ser implementará el procedimiento previo con el siguiente ejemplo.

🛞 🍈 🛅 🗾 🗾 pi@raspberrypi: ~/D	Pictures	💭 🖇 🍢 📣 22:47
	pi@raspberrypi: ~/Desktop/Practicas	~ = x
File Edit Tabs Help		
GNU nano 7.2	e52.s	
.text .global_start _start: mov r0,#5 mov r1,#0x01 subs r3,r0,r1		
bed igual bne diferente igual: mov r4,#1 b fin		
diferente: mov r4,#0 fin: mov r0,r3 ldr r1,=resultado str r0,[r1]		
final: b final		
resultado: .word 0	[Read 21 lines]	
AG Help AO Write Out AW Where Is AX Exit AR Read File AN Replace	VK Cut AT Execute AC Location M-U Un AU Paste AJ Justify A/ Go To Line M-E Re	do M-A Set Mark do M-6 Copy

Figura 9. Código ejemplo

Generando el código ejecutable.

pi@raspberrypi:~/Desktop/Practicas \$ as -g -o e52.o e52.s
pi@raspberrypi:~/Desktop/Practicas \$ ld -o e52 e52.o

Figura 10. Proceso de ensamblado y ligado

Notar que en el proceso de ensamblado se ha agregado el parámetro **-g** para habilitar requerimientos de gdb.

Para conocer el código generado, se recomienda revisar el contenido del programa objeto *.o con el comando:

objdump -s -d NombrePrograma.o

Contents of section .text:		
0000 0500a0e3 0110a0e3 013050e0 (0000000a	0P
0010 0100001a 0140a0e3 000000ea	0040a0e3	@
0020 0300a0e1 04109fe5 000081e5	fefffea	
0030 00000000		
Contents of section .data:		
0000 00000000		
Contents of section .debug_line:		
0000 3b000000 03001c00 00000201	fb0e0d00	;
0010 01010101 00000001 00000100	6535322e	e52.
0020 73000000 00000005 02000000	00152f2f	s//
0030 2f2f2f2f 2f2f2f2f 2f2c0202	000101	////////
Contents of section .debug_info:		
0000 22000000 02000000 00000401	000000000	"
0010 00000000 34000000 00000000	06000000	4
0020 21000000 0180		
Contents of section .debug_abbrev		
0000 01110010 06110112 01030e1b	0e250e13	%
0010 05000000		
Contents of section .debug_aranges	s:	
0000 1c000000 02000000 00000400	000000000	
0010 00000000 34000000 00000000	000000000	4
Contents of section .debug_str:		
0000 6535322e 73002f68 6f6d652f	70692f44	e52.s./home/pi/D
0010 65736b74 6f702f50 72616374	69636173	esktop/Practicas
0020 00474e55 20415320 322e3430	00	.GNU AS 2.40.

```
Contents of section .ARM.attributes:
0000 41110000 00616561 62690001 07000000 A....aeabi.....
0010 0801
Disassembly of section .text:
00000000 <_start>:
       e3a00005
                      mov
                              r0, #5
  0:
       e3a01001
                      mov
       e0503001
                      subs
       0a000000
                      beq
                              14 <igual>
 10:
       1a000001
                      bne
                              1c <diferente>
00000014 <igual>:
       e3a04001
 14:
                      mov
                              r4, #1
 18:
       ea000000
                              20 <fin>
0000001c <diferente>:
       e3a04000
                      mov
                              r4, #0
00000020 <fin>:
     e1a00003
                      mov
 24:
       e59f1004
                              r1, [pc, #4] @ 30 <final+0x4>
       e5810000
                              r0, [r1]
0000002c <final>:
       eafffffe
                              2c <final>
 2c:
       000000000
                              0×00000000
                       .word
```

Figura 11. Salida código objeto

En caso de ejecutar el programa, no genera salida a la consola y se mostrará el contenido de la figura 12.



Figura 12. Ejecución

C. GDB GNU Debugger.

Se invocará al GNU debugger escribiendo: gdb NombrePrograma

pi@raspberryi:~\$ gdb e52



Figura 13. Prompt del debugger gdb

Aparecerá el entorno mostrado en la figura 13; notar que el debugger queda listo para recibir comandos mediante su *prompt*: (*gdb*).

Comandos **gdb** más usados.

Comando	Abreviación	Descripción	Ejemplo
break	b	Establece un breakpoint.	b 5
run	r	Ejecución del programa en un solo paso o hasta encontrar un break point.	r
list	1	Despliega código.	1
info registers	ir	Despliega información de registros.	lr
desassemble	disas	Muestra código desensamblado.	disas
stepi	stepi	Ejecuta una o más instrucciones.	stepi stepi 10
continue	cont	Continua la ejecución de un programa de manera continua, hasta encontrar otro breakpoint; en caso de quedar en un ciclo infinito, salir con Ctrl-C.	С
examine memory /nfs addess	x /nfs address	Examinaelcontenidodememoria; donde:nNúmero localidades.FFormato de despliegue:oOctalxHexadecimaldDecimaluUnsigned dec.tBit.fFloat.aAddress.hHalfword.wWord.gGiant (8 bytes).sSizebByte.hHalfword.wWord.gGiant (8 bytes).Address.	x/4xw 0x1000
quit	q	Salir de gdb.	q

Tabla 1. Comandos gdb
Ejemplos de uso:

a. Listar el programa.



Figura 14. Comando I

b. Establecer breakpoint.



Figura 15. Comando b

c. Ejecutar hasta el breakpoint.

(gdb) r Starting program: /home/pi/Desktop/Practicas/e52 Breakpoint 1, _start () at e52.s:4 4 __start: mov r0,#5

Figura 16. Comando r

d. Desplegar contenido de registros.

(gdb) i r		
r0	Θ×Θ	Θ
r1	0×0	Θ
r2	0×0	Θ
r3	0×0	Θ
r4	0×0	0
r5	0×0	0
r6	0×0	0
r7	0×0	0
r8	0×0	0
r9	0×0	Θ
r10	0×0	Θ
r11	0×0	Θ
r12	0×0	Θ
sp	0xfffef100	0xfffef100
lr	0×0	Θ
рс	0x10074	0x10074 <_start>
cpsr	0×10	16
fpscr	0×0	Θ
tpidruro	<unavailable></unavailable>	
(gdb)		

Figura 17. Comando i r

e. Desplegar programa desensamblado.

Dump of	assen	ibler code	for fun	ction	st	art:
=> 0x00		<+0>;	mov			
		<+4>;	mov			
		<+8>:	subs			
		<+12>:				<igual></igual>
		<+16>:	bne			<diferente></diferente>



f. Desplegar programa desensamblado indicando inicio y final.

g.

(gdb) disas 0x10074, 0x100a0			
Dump of assembler code from	10074 to		
=> 0x00010074 <_start+0>:			
<pre>exceeding?# <_start+4>;</pre>			
<pre>exd001007c <_start+8>:</pre>			
<pre>@web010080 <_start+12>:</pre>	beq		<igual></igual>
8x800111934 <_start+16>:			<diferente></diferente>
<pre>()x000][0000] <igual+0>:</igual+0></pre>			
Dx0001000n <igual+4>:</igual+4>			<fin></fin>
<pre>9x00010000 <diferente+0>:</diferente+0></pre>			
0x00010001 <fin+0>: mov</fin+0>			
Dx00010000 <fin+4>: ldr</fin+4>	r1, [pt	,]	@ 0x100a4 <final+4></final+4>
Byond 1000c <fin+8>: str</fin+8>	ro. Fri	1	

Figura 19. Comando disas dir_inc dir_fin

h. Ejecutar una instrucción.



Figura 20. Comando stepi

i. Mostrar resultado de la ejecución.



Figura 21. Comando i r

j. Continuar ejecución continua del programa.

(gdb) cont	
Continuing.	
^C	
Program received signal SIGI	NT, Interrupt.
final () at e52.s:15	
15 final: b final	

Figura 22. Comando cont

k. Desplegar registros.

(adb) i r		
(905) 1 1	0×4	1
10	0,44	4
rı	0x110a8	69800
r2	Θ×Θ	Θ
r3	0×4	4
r4	0×0	Θ
r5	0×0	Θ
r6	0×0	Θ
r7	0×0	Θ
r8	0×0	Θ
r9	0×0	Θ
r10	0×0	Θ
r11	0×0	Θ
r12	0×0	Θ
sp	0xfffef100	0xfffef100
lr	0×0	Θ
рс	0x100a0	0x100a0 <final></final>
cpsr	0x20000010	536870928
fpscr	Θ×Θ	Θ

Figura 23. Comando i r

I. Mostrar memoria.

(gdb) x /8xw	0x110a8			
	0x00000004	0x00001141	0x61656100	0x01006962
	0×0000007	0x00000108	0x0000001c	0x00000002

Figura 24. Comando x

m. Salir de gdb.

Comando: **q** (quit)

5

Raspberry Pi

Entorno de Desarrollo Integrado Code::Blocks

El IDE Code::blocks permite el uso del paquete GNU GCC Compiler para escribir, compilar y ejecutar programas escritos en ensamblador dentro de la plataforma Raspberry.

I. Instalación.

Una vez corriendo el sistema operativo en la Raspberry, se requiere de realizar lo siguiente:

a. Actualizar los paquetes:

sudo apt-get update

b. Instalar Code::Blocks

sudo apt-get install codeblocks

II. Abrir Code:Blocks.

Una vez concluido el proceso de instalación, encontrará al IDE Code::Blocks en el grupo de programación.



Figura 1. Ejecución Code::Blocks



Figura 2. Etapa de arranque

El proceso para desarrollar programas en ensamblador consiste en:

- 1. Creación del proyecto.
- 2. Eliminar el archivo main.c del proyecto.
- 3. Agregar el archivo en ensamblador **nom_prog.s**.
- 4. Compilar el programa.
- 5. Ejecución del programa.

1. Crear proyecto nuevo.



Figura 3. Creación de proyecto nuevo

1.1. Configurar proyecto.

File Edit View Search	Project Build E	Debug Tools Plu	gins Settings	Start here - Code:Bloc Help	ks 20.03	• VE (e N z e	× II 0 0 0
	*						*	
Projects	Start here 🗷			New from temps	ite		~ ^ X	
Workspace		Build targets Files Custom User templates	Category: -All ARM Project AVR Project Fro, ct	categories>	Empty project Empty project ETK project EDK Fortran DLL	Fortran Fortr GLFi	Cancel View as • Large icons	32 bit if the Day
		TIP Try right clic 1. Select a wizar 2. Select a speci 3. Press Go	king an item d type first on the fo wizard from th	e left ne mein window (filter () () () () () () () () () () () () ()	by categories if pme/pi/asm/p pme/pi/asm/p pme/pi/asm/p	needed) (4/p4.cb) (3_1/p3_ (2_1/p2_	p 1.cbp 1.cbp	eature

Figura 4. Seleccionar Console application

1.2. Seleccionar el lenguaje de programación.

Será utilizado el compilador de C, por lo tanto, seleccionar lenguaje C; entonces presionar Next.

	10-1	Start Here - Gode. Dio	Shot home Code Blocks 20.03			0 10 10.55
le Edit View Search	Project Build Debug	Tools Plugins Settings	Help			
				I IN SERVICE AT LE	a waa a a ma	
	-		Cossole application	× ^ ×		
naderseur III)	Start here					
rojects		Console	lease select the language you w	ant to use		
😡 Workspace		1	Please make a selection	1		
			¢)			
			C++			
					tode - 32 bit	
					Long Martin	
					Tip of the Day	
					ew feature	
				T announced in concerns		
			< Back	Next > Cancel		
			🚺 🝵 /home/pi/a	sm/p3_1/p3_1.cbp		
			= /home/ni/z	sm/n2 1/n2 1 cho		

Figura 6. Configurar lenguaje

1.3. Nombrar el proyecto.

Indicar el nombre y ubicación del proyecto, continuar seleccionando Next.

8		>_		(pi)			-	Start	here -	Code	Blo.	-	Co	nsole applic	ation		S §	*	××	4 0)	17	18
ilo Edit	View	20	arch	Project	Build	Dehua	Tool		lucins	Sa	ttinae	Hol	200									1.7
	ich.		a	- Tojeo.	02 10	Debug	0	an B	inginta		in iga	1 100	er anti-			T ret	7					
1.3 649	100		-		55 (US)	12	-	-00		-	2	~	2.2			1 13	-					
Þ. «e	92	92	8-	-				0	CONISO	e app	Reatio	n				· ^	×					
		_			Cons	ole	Ple	ase	select	the fo	lder v	here	you	want the ne	ew proje	ect		_		*		
Drojecte			- CK				to l	be cri	eated	as we	ell as i	s title	ŧ.									
-Tujeeta							Pro	oject	title													
O WOR	kspace	e					P	ractio	cal_Ej	eq												
						1	Fol	lder t	o crea	te pro	ject in											
						1	1	nome	/pi/as	m/												
					-		Pro	oject	filenar	ne:												
							P	ractio	cal_Ej	e0.cb	p											
							Re	sultin	ng filer	ame								00.200	-	227.5		
							17	nome	/pi/as	m/Pr	actica	1_Eje	0/P	ractica1_Eje	0.cbp		1	ode -	32.0	ut.		
				-																		
																	- 3	Tip of	the [Day		
																		wfea	ature			
											×	Back		Next >	Ca	incel						
				er -																		
									1	/h	ome/	oi/asr	m/p	4/p4.cbp								
									10	i /h	ome/	oi/asr	m/p	3 1/p3 1.0	cbp							

Figura 7. Nombrar proyecto

1.4. Mantener la configuración por defecto, solo presionar Finish.

File Edit View Search	[pi] Project Build Debug	Start here - Code:Blo Start here - Code:Blo Tools Plugins Settin	la <mark>Fra</mark> Cor ocks 20.03 ngs Help	nsole applica	tion 🔓	17:18 × • • • • • • • • • • • • • • • • • •
N 20 40 40 40		Console applie	ation		~ ~ `	×
Projects O Workspace	🐻 Console	Please select the com you want enabled in y Compiler: GNU GCC Compiler	piler to use a our project.	ind which co	nfigurations	•
		Create 'Debug' co 'Debug' options-	nfiguration:	Debug		
		Objects a day t day	bin/Debug/	6		
		✓ Create 'Release" c	onfiguration:	Release		iode - 32 bit
		"Release" options				The of the Days
		Output dir.:	bin/Release	e/		lip of the Day
		Objects output dir:	obj/Releas	e/		ew feature
			< Back	Finish	Cancel	
		/hon	ne/pi/asm/p	4/p4.cbp 3 1/p3 1.ct	q	
						default

Figura 8. Concluir configuración del proyecto

La creación del proyecto genera el programa fuente de tipo C, con la plantilla mostrada en la figura 9.

👅 🕘 🛑 💌 📗	[pi] 🕂 main.c [Practica1_Eje 📝 🔧 🍾 📣 17:18
File Edit View Search	Project Build Debug Tools Plugins Settings Help
j enionals Management	
Projects	main.c 🛞
Workspace Practica1_Eje0 Sources main.c	<pre>3 4 5 6 7 6 7 8 9 1 7 8 9 1 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7</pre>
/ho U	x (LF) UTF-8 Line 1, Col 1, Pos 0 Insert Read/Write default

Figura 9, Proyecto completo

2. Eliminar programa main.c.

El programa main.c debe de ser eliminado, para ser intercambiado por el programa en ensamblador; presionar el botón secundario en el archivo main.c y seleccionar eliminar.

3. Agregar archivo fuente *.s.

Seleccionar File y nuevo para agregar al proyecto; hará la solicitud mostrada en la figura 10.

File Edit View Searc	[pi] ch Project Build	*Untitled1 - Code:Blo *Untitled1 - Code:Blocks 20.03 Debug Tools Plugins Settings Help	
	• [*
Terrigeroons Torrigeroon Tor	Do you	Add file to project •• want to add this new file in the active project (has to be saved fir No	st)? Yes
/ho	Unix (LF)	UTF-8 Line 1, Col 1, Pos 0 Insert	Modified Read/Write default

Figura 10. Confirmación al agregar el programa fuente al proyecto

Indicar el nombre del programa; este debe tener la extensión **.s**; salvar.

کا 👘 🕕 🌔	[pi]	Hontitled2 - Code::Blo	Save file	∦ *	1 7:20
		Save file			× ^
lame: practica1_ej	(0.s				
Home	< ✿ pi asm	Practica1_Eje0 +			C
🖿 Desktop	Name			Size T	ype Modifie
Documents	🔟 main.c			99 bytes	ext 17:18
Downloads					
Ø Music					
D Pictures					
 Videos 					
Practica1_Eje0					
+ Other Locations					
			C/C++ files		,



۵ 🛑 🐚 👅	pi]	practica1_ej0.s - Cod	🕂 Multiple selection	7:20
		practica1_ej0.s - Code_Bl	ocks 20.03	~ = ×
File Edit View Search P	roject Build Deb	oug Tools Plugins Setting	s Help	
🕒 🕒 🗇 💊 🖻		🔍 🔍 🚺 😰 🕨 🛸	📀 🛄 Debug 🔻	
► 48 90 M 20 90	× n 🖂	× 🛛		
	÷			
Management (20	practica1_ej0.s	9		
Projects	1			
🔻 🙆 Workspace	2	Multiple selectio	n × • ×	
Practica 1_Eje0	Select 1	he targets this file should belo	na to:	
	🖌 Det	pug	Wildcard select	
	🗷 Rel	ease	Togale selection	
			Select All	
			Deselect All	
			Selected: 2	
			_ Selected, 2	
		C	Cancel OK	
/ho Unix	(LF) UTF-	8 Line 1, Col 1, Pos	0 Insert	Read/Write default

Figura 12. Mantener la configuración inicial

Se abrirá el entorno para edición del programa fuente.

ا کا 🕒 🌔 🐌	[pi]	eje1.s [eje1] - Code::Bl	local code plast a ph co		₹ © \$	e d) 16:37
File Edit View Search	Project Build Debug	g Tools Plugins Settings Help	Debug 🔹 📑			1
	· •			*		
Projects	eje1.s 🗵					
 ♥ Workspace ♥ ■ cje1 						



El siguiente paso, será la escritura del programa en ensamblador.



Figura 14. Programa fuente

4. Compilar el proyecto.

Figura 15. Proceso de compilación

En caso de no existir error, quedará listo para la ejecución, se requiere habilitar un break point en la primera instrucción del programa (colocarse del lado derecho del número de línea de esta, aparecerá un círculo rojo (³ main: mov R0,#0</sup>).

File Edit View Search Project	practica _cj0.s [Practica _cja0] - Code: Blocks 20.03 Build Debug Tools Plugins Settings Help			~ = ×
File Edit View Search Project	Build Debug Tools Plugins Settings Help			
	🗈 🙀 🔍 🥾 🧼 👁 🖬 Debug			
+		▼ 13		
No we as we as as w				
Management 🐵 👔	1			0 1
Projects	ar_eju.s ta			
✓ Workspace ✓ Morkspace ✓ Markspace ✓ Marksp	<pre>.global main main: mov R0, #0 mov r2, #0 mov r2, #0 loop1; hadd r0, r0, #1 cmp r1, r0 bne loop1 loop2: add r0, r0, #-1 cmp r2, r0 beq loop1 b loop2</pre>			
/ho Unix (LF)	UTF-8 Line 6, Col 9, Pos 97 Inser	t Read/	Write defaul	t

Figura 16. Ubicación del break point

5. Ejecución.

Presionar el comando (Debug/Stop), para iniciar la ejecución del programa; Code::Block genera las pantallas para ejecución por pasos, mostrará parte del código la ventana del desensamblador, que será actualizado conforme se avance en el proceso.

🛎 🍈 🐂 🔜 📔	[pi] Hactica	a1_ej0.s (Pract)	Program Console	20 🖹 🍾 📣 17:43
	practica 1_ei0 s (Pra	ctica 1_Eie0] - Code:E	locks 20.03	~ ¤ ×
File Edit View Search	Project Build Debug Tools Plu	ains Settings Help	1	
· · · · · · ·	· · · · · · · · · · · · · · · · · · ·	0 × % 0 (🕽 Debug 👻 Li	► <u>₩</u> ≪ ×
	*			-
Henspersent 30	CFU Requiters	190	Distanty	0
Projects	Regist Hex Interpreted		Function:	
 Workspace Practica 1_Eje0 ASM Sources practica 1_ej0. 	P0 9K1 1 1 8x7eff944 2130704804 r2 9x7eff79ac 2130704812 r3 9x1936 95512 r4 9x19378 95552 r5 9x0 0 r6 9x1926 95522 r7 9x0 0 r8 9x0 0 r8 9x0 0 r9 9x0 0 r10 9x7ff7000 1205494503 r11 9x0 0		Frame start Ox7eH1848 exclose mov r9, v9 exclose mov r2, v9 exclose mov r2, v9 Mixed Mode	Adjust Save to text file
	Address: 0x0 (e.g. 0x401060, or &variable, or \$\$ee Cannot access memory at address 0x8	ix)		Bytes: 32 - Go
/ho Un	x (LF) UTF-8 Line	3, Col 1, Pos 35	Insert	Read/Write default

Figura 17. Estado inicial del *Debugger*

Conociendo la dirección de asignación del programa, podrá ingresarla para ver su comportamiento durante la ejecución (Address: 0x103d0).

	processo (celora processo) - decisio	
File Edit View Searc	n Project Build Debug Tools Plugins Settings Help)
C 🕒 🖉 🚺		🗅 Debug 🛛 👻 🐻 🕨 😽
	· ·	*
	10 CPU Repútare 10	Disastaninity
Projects	Regist Hex Interpreted	Function:
 Practical_Eje0 ASM Sources practical_e 	r2 6x7erff9ac 2180704812 r3 0x10000 66312 r4 0x10010 66312 r5 0x6 8 r6 0x1000 6832 r7 0x6 8 r7 0x6 8 r7 0x6 8 r9 0x6 8	SELONDE 100V 72. HO
	r18 6x76ff690 1996484688 r11 040 0	Adjust Save to text file
	Address: 0x103d0 (e.g. 0x401060, or &variable, or SSeax)	Bytes 64 💌 Go
	[enclosede: eo eo acia con to acial en a care to en e ez, 3 exclose: eo eo sa et a f. f. ff fialet en exe ez eo en ze acia,eo (40, 20, 20, 20, 20, 20, 20, 20, 20, 20, 2	a. apsină ,68rmă èderé.p.ă #rficială. ficană. rficană

Figura 18. Estado actual de los registros y memoria

Presionar 🧏, para ejecución del programa por pasos y así comprobar el funcionamiento del programa.

	*		*	
insermit (2	CPU Registers		H Dussembly	
Projects	Regist Hex Inte	erpreted	Function:	
 Workspace Fractica1_Eje0 ASM Sources practica1_ej0. 	ré esté 0 r1 exis 9 r2 exis 0 r3 oxtabás 6651 r4 oxtabás 6651 r5 oxtabás 6651 r6 oxtabás 6651 r8 exis 0 r8 exis 0 r18 exis 0 r10 oxtreff'ese 1965 r11 exis 0	12 52 72	Frame start Ox7eff/848 betage day rd, rd, rd exteed orp rd, rd exteed be extende -toopt-	text file
	Address: 0x0 (e.g. 0x401060, or a carriot access serving at	&vanable,or\$\$eax) t address exe	Bytes: 32 💌	Go
			Close b	11

Figura 19. Resultado de la ejecución

6

CPUlator

Simulador en línea

Es una herramienta útil para la escritura y depuración en línea de programas en ensamblador, aplica para diferentes versiones de arquitecturas. Será de interés las arquitecturas ARM, son asignadas las primeras direcciones de memoria para ubicar el código de programa y los datos.

El acceso al simulador lo puede realizar desde cualquier navegador, escribiendo cpulator; se selecciona la opción:

https://cpulator.01xz.net/

Aparecerá la pantalla de inicio en la que se seleccionará la arquitectura y el sistema a utilizar.

architecture	System	
Any	 Nios II generic 	4
Nios II	Nios II DE1-SoC	
ARMv7	Nios II DE1-SoC (v16.1)	
MIPS32r5	Nios II DE2-115	
MIPS32r6	Nios II DE2-115 (v16.1)	
RISC-V RV32	Nios II DE2	
	Nios II DE0	
https://cpulator.01xz.n	et/?sys=nios	Go
Nios II generic		
Nios II generic	norv and no other I/O devices	



Raspberry Pi está diseñada usando la arquitectura **ARM**; por lo que se podrán simular programas en ensamblador seleccionando el sistema genérico de la arquitectura **ARMv7**.

Architecture		System	
Any Nios II ARMv7 MIPS32r5 MIPS32r6 RISC-V RV32		ARMv7 generic ARMv7 DE1-SoC ARMv7 DE1-SoC (v16.1) Nios II generic Nios II DE1-SoC Nios II DE1-SoC (v16.1) Nios II DE2-115	Î
https://cpulator.01xz.r	net/?sys=arm	í.	Go
ARMv7 generic ARMv7 system with 4 GB of me	mory and no othe	r I/O devices	

Figura 2. Selección; arquitectura ARM7

I. Descripción CPUlator

Una vez seleccionada la arquitectura, se presenta el entorno de trabajo.

Basistara	42 0es40	9840	and the second se			N.CIODU	Tile	Malaca			
		and a strength		- F4	C#-2	CHERRIS	onen	melba-	 	 	
efresh		Compile a	erri-E) and Load (F5)	Language	ARMv7 ~	untitled.s					
rD eek r1 eek r2 eek r3 eek r5 eek r5 eek r6 eek r7 eek r8 eek r6 eek r8	ebende eb	a stor	l start ti								
mber Display	Options	8									
		Ø Editor (C	WEI 4 Ditas	sembly (Ctri	-D) Q Memi	ory (Ctrl-Mb					

Figura 3. Entorno CPUlator

Se identifican las distintas zonas del entorno; entre los que se encuentran:

- 1. Control de simulación.
- 2. Área de edición.
- 3. Área de desensamblado.
- 4. Área de memoria.
- 5. Área de registros.
- 6. Área de configuración.
- 7. Área de mensajes de salida.

1.- Control de simulación. En la parte superior se encuentran los íconos para control del proceso de simulación y el manejo de archivos.

Stopped	Step Into	Step Over	Step Out	Continue	Stop F4	Restart	Reload	File -	Help 🔻
	P2	CUI-F2	3111(1-F2	rs .	F4	CIN-K	CERESTINGE		

Figura 4. Control de simulación

2.- Área de edición. Ubicar en la zona del editor para escribir un programa, de manera automática aparece el formato del programa en ensamblador. El simulador no requiere la directiva *.text* (*inicio del código*); en caso de usar *datos* deberá incluir la zona de ellos, mediante *.data*.



Figura 5. Editor CPUlator

3.- Área de desensamblado: Mostrará el programa en el formato original, la construcción en formato pseudoensamblador, las direcciones de memoria donde el entorno asignará a las instrucciones y los códigos de las instrucciones; además dará seguimiento al proceso de simulación.

</th <th colspan="8">V Disassembly (Ctrl-D)</th>	V Disassembly (Ctrl-D)							
Go to address, label, or register:			ter: 000000	90	▼ Refresh			
•	Address	Opcode	Disassembly	1				
	ffffffe8	aaaaaaaa	bge	0xfeaaaa98				
	fffffec	aaaaaaaa	bge	0xfeaaaa9c				
	ffffff0	aaaaaaaa	bge	0xfeaaaaa0				
	fffffff4	aaaaaaaa	bge	0xfeaaaaa4				
	ffffff8	aaaaaaaa	bge	0xfeaaaaa8				
	ffffffc	aaaaaaaa	bge	0xfeaaaaac				
	00000000	aaaaaaaa	bge	0xfeaaaab0				
	00000004	aaaaaaaa	bge	0xfeaaaab4				
	00000008	aaaaaaaa	bge	0xfeaaaab8				
	0000000c	aaaaaaaa	bge	0xfeaaaabc				
	00000010	aaaaaaaa	bge	0xfeaaaac0				
	00000014	aaaaaaaa	bge	0xfeaaaac4				
	00000018	aaaaaaaa	bge	0xfeaaaac8				
	0000001c	aaaaaaaa	bge	0xfeaaaacc				
	00000020	aaaaaaaa	bge	0xfeaaaad0				
	00000024	aaaaaaaa	bge	0xfeaaaad4				
	00000028	aaaaaaaa	bge	0xfeaaaad8				
	0000002c	aaaaaaaa	bge	0xfeaaaadc				
	Editor (Ctrl-I	E) 💔 Disass	embly (Ctrl-D)	Q Memory (Ctrl-M)				

Figura 6. Ventana del desensamblador

5. Área de Memoria. Podrá ver y modificar el contenido de memoria.

6.

Q Memory (Ctrl-M)										
Go to addre	Go to address, label, or register: 📃 👻 Refresh									
Address	Memory co	ntents and A	SCII							
00000000	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	••••					
00000010	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
00000020	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
00000030	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
00000040	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
00000050	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
00000060	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
00000070	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
00000080	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
00000090	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
000000a0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
000000b0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
000000c0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
000000d0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
000000e0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
000000f0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
00000100	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••					
00000110	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	••••					
😰 Editor (Ctrl-E) 🌗 Disassembly (Ctrl-D) 🔍 Memory (Ctrl-M)										

Figura 7. Despliegue y control de memoria

5.- Área de registros. Permite visualizar y modificar el contenido de los registros de la arquitectura.

·			
	0000000 0000000 0000000 0000000 0000000	00000000 00000000 00000000 00000000 0000	00000000 00000000 00000000 00000000 0000

Figura 8. Registros

6.- Área de configuraciones. Configura opciones de despliegue, simulación y manejo de datos.

🟦 Registers	🙀 Call stack	👬 Trace	Settings	
飛 Breakpoin 飛 Counters	ts 🔒 🛣 Watchp	ooints 🟦 Symbols	Number Display Options	Î
			Format: Hexadecimal	-

Figura 9. Configuración del IDE

7.- Zona de mensajes. Mostrará información del proceso de compilado.

⊠ Messages	
CPUlator has started with system ARMv7 generic containing a ARMv7 processor.	-
	÷

Figura 10. Ventana de mensajes

II. Uso de CPUlator.

El proceso consta de:

- A. Escritura del programa.
- B. Compilar el programa.
- C. Simular el programa.
- D. Revisar la ejecución.

A. Escritura del programa.

Siguiendo la estructura del programa en ensamblador, usar el editor de CPUlator.

```
Editor (Ctrl-E)
 Compile and Load (F5)
                         Language: ARMv7 ~
                                             untitled (15).s [changed since compile]
 1 .global _start
 2 _start:
 3
 4 main: ldr r0,=dato1
          ldr r0,[r0]
 5
 6
          ldr r1,=dato2
          ldr r1,[r1]
 7
          add r2,r1,r0
 8
 9
          ldr r3,=resultado
          str r2,[r3]
10
          b main
11
12 .data
13 dato1: .word 9
14 dato2: .word 9
15 resultado: .word 0
16
C Editor (Ctrl-E) </>
Disassembly (Ctrl-D) Q Memory (Ctrl-M)
```

B. Compilar el programa.

Compile and Load (F5)

Figura 10. Botón para compilar

Al finalizar el proceso de compilado, generará el estado de mismo.

Messages

Assemble: arm-eabi-as -mfloat-abi=softfp -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmk3YsIl.s.o work/asmk3YsIl.s

Link: arm-eabi-ld --script build_arm.ld -e _start -u _start -o work/asmk3YsIl.s.elf work/asmk3YsIl.s.o

Compile succeeded.



De manera automática mostrará el programa en el área de desensamblado y quedará listo para iniciar la simulación.

<	Disassembly (Ctrl-D)						
Go	o to address,	label, or regist	er: 00000000 • Refresh				
•	Address	Opcode	Disassembly				
		1	.global _start start: main: ldr r0,=dato1 _start:				
	00000000 00000004	e59f0018 e5900000	ldr r0, [pc, #24] ; 0x20 idr r0, [r0]				
	00000008 00000000c 00000010	e59f1014 e5911000 e0812000	ldr r1, [pc, #20] ; 0x24 dr r1, [r1] add r2, r1, r0				
	00000014 00000018	e59f300c e5832000	ldr r3,=resultado ldr r3, [pc, #12] ; 0x28 str r2, [r3]				
	0000001c	eafffff7	1 b main b 0x0 (0x0: main) main: ldr r0,=dato1				
	00000020	00000030	andeq r0, r0, r0, LSR r0 ldr r1,=dato2				
	Ctrl-E) Ctrl-E) Ctrl-D) Q Memory (Ctrl-M)						

a. Área de código

🕱 Registers			Isassembly (Ctrl-D)					
Refresh		1	Go to address,	ster: 00000000 - Refresh				
r0	0000009		Address	Opcode	Disassembly			
r1	00000009		00000020	00000030	andeo r0, r0, r0, LSR r0			
r2	00000012			1000000000	6 ldr r1.=dato2			
F3	0000038		00000024	00000034	andeg r0, r0, r4, LSR r0			
14	80000000				9 ldr r3.=resultado			
r6	999999999		00000028	00000038	andeq r0, r0, r8, LSR r0			
17	0000000		0000002c	000000000	andeq r0, r0, r0			
r8	00000000				dato1:			
r9	0000000		00000030	00000009	andeq r0, r0, r9			
r10	00000000				dato2:			
r11	0000000	Y	00000034	00000009	andeq r0, r0, r9			
Registers	🟦 Call stack 🙀 Trace				resultado:			
# Breakpoin	ts # Watchpoints # Symbols		00000038	00000012	andeq r0, r0, r2, LSL r0			
Counterr	I AN THE THE PARTY IN THE		0000003c	00000000	andeq r0, r0, r0			
TE Counters					_end:			
Settings Number Display Options			00000040	aaaaaaaa	bge 0xteaaaat0			
			00000044	aaaaaaaa	bge Oxteaaaat4			
			00000048	aaaaaaaa	bge 0xteaaaat8			
Size: Word			0000004C	aaaaaaaa	bge exteasaatc			
Size: Word V			00000050	aaaaaaaa	bge Uxteaaab00			
Format: Hexadecimal 🗸 🚽			Z Editor (Ctrl-	E)	sembly (Ctrl-D) 🔍 Memory (Ctrl-M)			

b. Área de datos

Figura 13. Programa ensamblado

Así mismo podrá ver el contenido de memoria; se identifican los códigos de las instrucciones y el área de datos.

Q Memory	Q Memory (Ctrl-M)								
Go to addre	Go to address, label, or register: 📃 👻 Refresh								
Address	Memory con	ntents and A	SCII						
00000000	e59f0018	e5900000	e59f1014	e5911000	••••				
00000010	e0812000	e59f300c	e5832000	eafffff7	• •• •0•• • •• •••				
00000020	00000030	00000034	00000038	00000000	0••• 4••• 8••• ••••				
00000030	00000009	00000009	00000000	00000000	•••• •••• ••••				
00000040	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••				
00000050	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••				
00000060	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• ••• ••••				
00000070	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• ••• ••••				
00000080	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• ••• ••••				
00000090	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••				
000000a0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• ••• ••••				
000000b0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••				
000000c0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••				
000000d0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••				
000000e0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••				
000000f0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••				
00000100	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••				
00000110	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••				
00000120	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa	•••• •••• ••••				
🕼 Editor (Ctrl-E) 💔 Disassembly (Ctrl-D) 🔍 Memory (Ctrl-M)									

Figura 14. Contenido de memoria

C. Simulación.

Para iniciar el proceso de simulación presionar ; podrá ver la ejecución paso a paso, con lo que se comprobará el funcionamiento del programa.

Stopped	Step Into F2	Step Over Ctrl-F2	Step O Shift-F	ut Continue 2 F3	Stop F4	Restart Ctrl-R	Reload Ctrl-Shift-L	File 🕶	He	lp 🖛
🟦 Registers				V Disassen	nbly (Ctrl-D)					
Refresh			â	Go to address	, label, or regi	ster: 000000	000		-	Refresh
r0	00000009			Address	Opcode	Disassembl	у			
r2 r3 r4	00000012 00000038 000000000					4 main: _start: main:	ldr r0,=da	tol		
r5 r6	000000000			00000000 00000004	e59f0018 e5900000	ldr 5 ldr	r0, [pc, r0, [r0]	#24]	0x2	9
r8 r9 r10	000000000000000000000000000000000000000			00000008 0000000c 00000010	e59f1014 e5911000 e0812000	ldr 7 ldr 8 add	r1, [pc, r1, [r1] r2, r1,	#20] ; r0	0x24	4
r11 Registers	Call stack	兼 Trace bints 兼 Syml	bols	00000014 00000018	e59f300c e5832000	9 ldr r3 ldr 10 str	r3, [pc, r2, [r3]	9 #12] ;	0x2	3
R Counters				0000001c	eafffff7	b Ox	:0 (0×0: m	ain)		
Settings			*	00000020	00000030	4 main: andeq	tdr r0,=da r0, r0,	tol r0, LSR r	0	
Size: Word	play Options	5	0	00000024	00000034	6 ldr r1 andeq	.,=dato2 r0, r0,	r4, LSR r	0	
Format: Hexadecimal 🗸			Editor (Ctrl	-E) 🏼 🔸 Disas	sembly (Ctrl-D) Q Memory	(Ctrl-M)			

Figura 15. Ejecución del programa; actualización de registros.

D. <u>Revisión de ejecución</u>.

Es recomendable revisar el comportamiento del programa comprobando en memoria los resultados obtenidos.

Stopped	Step Into F2	Step Over Ctrl-F2	Step O Shift-F	Dut Contin 2 F3	nue Stop F4	Restar Ctrl-R	rt Reloa Ctrl-Shift	d File -	Help	-		
🟦 Registers				Q Memor	y (Ctrl-M)							
Refresh			â	Go to addre	ess, label, or re	gister:			- R	efresh		
r0	0000009			Address	Memory co	ntents and A	ASCII					
r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11	000000012 00000038 00000000 00000000 00000000 00000000			00000000 00000020 00000020 00000030 00000040 00000050 00000060 00000070 00000070 00000070	e59f0018 e0812000 00000030 00000009 aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa	e5900000 e59f300c 00000034 00000009 aaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa	e59f1014 e5832000 00000038 06000012 aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaa	e5911000 eafffff7 00000000 aaaaaaa aaaaaaaa aaaaaaaa aaaaaa	0	0.4	8	
 飛 Registers 飛 Breakpoint 飛 Counters ふ Settings Number Dis Size: Word 	兼 Call stack s 兼 Watchpo play Options	兼 Trace bints 兼 Sym	bols	000000000 000000000 00000000 00000000 0000	22222222 222222222 22222222 22222222 2222		22222223 22222223 22222223 22222223 222222	aaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaa				
Format: Hexa	adecimal 🗸]	*	Editor (C	Ctrl-E) 4 Disa	assembly (Ctr	1-D) Q Men	nory (Ctrl-M)				

Figura 16. Ejecución de programa; revisión de memoria

7

Raspberry Pi Pico

Thonny - Micropython

Existen varias alternativas de programación de la plataforma Raspberry Pi Pico, entre ellas están C, C++, Arduino y Micropython.

Para la implementación del proyecto, se ha enfocado su desarrollo y propuesta empleando el IDE Thonny, programando en la versiónde Python, pensada en el uso de sistemas diseñadas con microcontroladores; conocida como Micropython.

Este trabajo indica el procedimiento para:

- Descargar el software Thonny.
- Instalación del IDE Thonny.
- Configuración del entorno de Thonny.
- Instalación del interprete para Micropython en la Raspberry Pi Pico.
- Editar, guardar y ejecutar programas.

Descarga de Thonny.

Para descargar el software Thonny, acceder al sitio oficial:

https://thonny.org/

Google	thonny	×	• @ Q	໑ 실 ⅲ	0 Î
	Tado Imégenes Videos Shapping Noticias Videos cortos Web Más		Herramientos		
	The Thonny neutribution on Traduct esta plana : Thonny, Python IDE for beginners Easy to get started. Thonny comes with Python 3,10 built in, so just one simple installer is needed and you're ready to learn programming. (You can also use a		Thonny Programa I		
	Version 3.1.1 > Thoriny, Python IDE for beginners.			a province and a second	
	Thonny and MicroPython Thonny 2.2 has plug-ins which add more general support for		hadrogy name a band("mot 's name 'name 's) and a band("mot 's) and an		
	Version 3.0.1 Version 3.0.1 Minimum supported Python version (both for the		Thonny es un entorno de desarrollo integrado gratuito		
	Search Bearch Results. Eesrch finished, found 22 page(s) matching the		y de código abierto para Py para principiantes. Fue crea programador estonio. Wikip	/thon que está diseñado sdo por Alver Annamaa, un pedia (Inglés) >	
	Thonny blog				

Figura 1. Ubicar el sitio de descarga de Thonny

Requiere especificar la versión del sistema operativos donde será instalado.



Figura 2. Selección del Sistema Operativo

Para cada sistema operativo existen diversas versiones, por lo que deberá elegir la requerida por su equipo de cómputo.

Thonny Download version 4.1.7 for Windows • Mac • Linux Official downloads for Windows Installer with 64-bit Python 3.10, requires 64-bit Windows 8.1 / 10 / 11 thonny-4.1.7.exe (21 MB) \Leftarrow recommended for you Installer with 32-bit Python 3.8, suitable for all Windows versions since 7 thonny-py38-4.1.7.exe (20 MB) Portable variant with 64-bit Python 3.10 thonny-4.1.7-windows-portable.zip (31 MB Portable variant with 32-bit Python 3.8 thonny-py38-4.1.7-windows-portable.zip (29 MB) Re-using an existing Python installation (for advanced users) pip install thonny

Figura 3. Selección de la versión de Thonny

Instalación de Thonny.

Una vez descargado, proceder a instalar (thonny-4.1.7.exe o la más actual disponible); seguir el proceso de instalación.

님 Setup - Thonny	- 🗆 X	💾 Setup - Thonny — 🗌 🗙
iess.py	Welcome to using Thonny!	License Agreement Please read the following important information before continuing.
1 import andom 2 n = ar randir	This wizard will upgrade Thonny to version 4.1.7 for your account.	Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.
<pre>guess = int(input while n != "guess while n != "guess f if guess < n: print("gu guess = i elif guess > print("gu guess = i guess = i elif guess = i guess</pre>		The MIT License (MIT) Copyright (c) 2024 Aivar Annamaa Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
4 print("yo	Next Cancel	I do not accept the agreement Back Next Cancel



Figura 4. Proceso de instalación

Uso y configuración de Thonny.

Al ejecutar Thonny, aparecerá el entorno de trabajo; en donde, se podrá identificar:

- El área de comandos, para manejo de archivos, configuraciones, botones de ejecución y paro del programa en curso.
- El área de edición, para escritura del código.
- Consola de Thonny, para ejecución en línea de comandos o despliegue de resultados e información.



Figura 5. Entorno Thonny

Configuración del interprete.

Seleccionar Micropython (Raspberry Pi Pico), en el Interprete dentro de las opciones de Thonny (figuras 6 y 7); en esta etapa no es requerida conectar la tarjeta a la computadora.

🄚 Thonny - <sin nomb<="" th=""><th>re> @ 1:1</th></sin>	re> @ 1:1
Fichero Editar Visualizar	Ejecutar Herramientas Ayuda
D 😂 🖬 🛛 🚸 🤉 3.	Configurar interprete

Figura 6. Configuración del interprete

Mantener la detección automática del puerto o REPL.

ieneral	Intérprete	Editor	Temas y Fuentes	Ejecutar y Depurar	Terminal	Consola	Asistente	
Dué ti	no de intér	nrete de	he utilizar Thonny	para elecutar su cód	lino?			
MicroP	ython (Rasp	oberry P	Pico)	para specarar su coo	.go:			~
Python	3 local							
Python	3 Remoto	(SSH)						
MicroP	ython (SSH)						
MicroP	ython (EV3)							
MicroP	ython (BBC	micro:b	it)					
MicroP	ython (Rasp	oberry P	Pico)					
MicroP	ython (RP2)	040)						
MicroP	ython (ESP3	32)						
MicroP	ython (ESP&	8266)						
MicroP	ython (gene	érico)						
CircuitP	ython (gen	érico)						
Puerto	O O WEDKE	PL.		3				
< Inte	enta detecta	ar el pue	rto automáticame	nte >				~
-								
⊻ Int	errumpir el	program	na de en funciona	miento al conectarse				
🗹 Sir	ncronizar el	reloj de	tiempo real del di	spositivo				
VU	ilizar la hora	a local er	n el reloj de tiempo	o real				
Re	iniciar el int	érprete	antes de eiecutar u	in script				
						Instala o a	ctualiza Mi	croPython
					1	actura e a	Contract ()	and function

Figura 7. Selección Micropython

Instalación del interprete Micropython.

Otro paso requerido es la instalación del interprete de Micropython en la tarjeta Raspberry Pi Pico; esto se puede realizar mediante dos opciones.

- Dentro del IDE Thonny
- Copiando el interprete directamente a la Raspberry Pi Pico.

Desde Thonny.

Conectar la tarjeta Raspberry Pi Pico a la computadora a través del cable USB-Micro_USB-B; en la ventana de configuración de Thonny y del puerto COM, del lado inferior – derecho, encontrará <u>Instala o actualiza MicroPython</u>.

	Consola Asistente
¿Qué tipo de intérprete debe utilizar Thonny para ejecutar su código?	
MicroPython (Raspberry Pi Pico)	2
Detalles	
Connecting via USB cable: Conecte su dispositivo a la computadora y seleccione el puerto correspon (buscar por un nombre de dispositivo, "USB Serial" o "UART"). Si no puede encontrarlo, es posible que primero deba instalar el controlad	diente a continuación or USB adecuado.
Connecting via WebREPL: If your device supports WebREPL, first connect via serial, make sure WebRI (import webrepl_setup), connect your computer and device to same network < WebREPL > below Puerto o WebREPL	EPL is enabled rk and select
< Intenta detectar el puerto automáticamente >	×

Figura 8. Instalar o actualizar Micropython

Seleccionar la versión de la tarjeta Raspberry Pico y las configuraciones requeridas.

🏗 Install or update MicroPython (UF2)	×	Th Install or update MicroPython (UF2)	×
Here you can install or update MicroPython for devices (this includes most boards meant for beginners).	having an UF2 bootloader	Here you can install or update MicroPython (this includes most boards meant for begin	for devices having an UF2 bootloader ners).
 Put your device into bootloader mode: some devices have to be plugged in while holding some require double-tapping the RESET button wi Wait for couple of seconds until the target volume a Select desired variant and version. Click "Install" and wait for some seconds until done. Close the dialog and start programming! 	the BOOTSEL button, th proper rythm. ppears.	 Put your device into bootloader mode: some devices have to be plugged in w some require double-tapping the RESE Wait for couple of seconds until the target Select desired variant and version. Click 'Install' and wait for some seconds u Close the dialog and start programming! 	hile holding the BOOTSEL button, T button with proper rythm. it volume appears. intil done.
Target volume RPI-RP2 (D:)	v	Target volume RPI-RP2 (D:)	~
family RP2		family RP2	
MicroPython family RP2	×	MicroPython family RP2	Y
variant	×	variant Raspberry Pi • Pico / P	ico H 👻
version	×	version 1.24.1	*
info [select one from 52 variants]		info https://micropython.o	rg/download/RPI_PICO
lins	talar Cancelar		Instalar Cancelar



En caso de no existir algún problema, indicará la conexión correcta, mediante el despliegue de la figura 10; la elección del puerto COM se ha realizado de manera automática.

The Thonny		1221	
Fichero Editar Visualizar Ejecutar Herramientas Ayuda			
		Asistente	
Consola		Puerto	
Process ended with exit code None.		, COM	
MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040 Type "help()" for more information.	Información del dispositivo		
MPY: soft reboot			
MPY: soft reboot MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040			
MPY: soft reboot MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040 Type "help()" for more information.		-	

Figura 10. Conexión completa

Transfiriendo RPI_PICO_20241129_v1.24.1.uf2 o la versión más reciente.

Descargar el programa correspondiente a la versión de su plataforma (Pico, Pico W, Pico 2 o Pico 2W); para la primera opción: RPI_PICO_20241129_v1.24.1.uf2, del enlace:

https://www.raspberrypi.com/documentation/microcontrollers/micropython.html

También puede descargar de:

https://micropython.org/download/RPI_PICO/

Para este procedimiento requiere de tres pasos.

- Presionar el botón BOOTSEL de Raspberry P Pico.
- Conectar el cable USB-Micro_USB-B entre la computadora y Raspberry Pi Pico (sin dejar de presionar el botón BOOTSEL), la tarjeta es reconocida como una unidad de memoria, en este momento podrá liberar el botón BOOTSEL (figura 11).
- Copiar el archivo RPI_PICO_20241129_v1.24.1.uf2 a Raspberry Pi Pico, la tarjeta queda lista para programar con Micropython; dejará de mostrarse como unidad de almacenamiento (figura 12).

Nombre	Fecha de modificación	Tipo
C INDEX.HTM	05/09/2008 04:20 p. m.	Microsoft Edge HTM
INFO_UF2.TXT	05/09/2008 04:20 p.m.	Documento de text

Figura 11. Detección de Raspberry Pi Pico

nclar al Copiar Pegar eso rápido Portapape	Ma Copiar ruta de acceso Pegar acceso directo des	Mover Copiar Eliminar Cambiar a a a nombre Organizar	Nueva carpeta Nuevo	Propiedades Abrir	Select Select	ccionar todo eleccionar ninguno rtir selección Seleccionar
-> - + 🛃 > D:	č.			ٽ ~	,O Busca	r en RPI-RP2 (D:)
 Fotos Público Este equipo 	Nombre	M 2.TXT		Fecha de mod 05/09/2008 0 05/09/2008 0	lificación 4:20 p. m. 4:20 p. m.	Tipo Microsoft Edge H Documento de te
 Descargas Documentos Escritorio 		-20241129-v1.24.1.uf2		20/03/2025 0	4:43 p. m.	Archivo UF2
 Imágenes Música Objetos 3D 	I					
Company of the second sec						

Figura 12. Interprete de Micropython instalado

Editar, guardar y ejecutar un programa.

Editar código.

Considerando la estructura de programación en Python, escribir el programa en la zona designada.

The Thonny - C\Users\ranay\Desktop\Raspberry 2024\Micropython\GPIO\Ejemplo\blink25.py @ 10:1	-	
Fichero Editar Visualizar Ejecutar Herramientas Ayuda		
LER ON CARP C		
blink25.py =	Asistente -	
<pre>import machine import utime LED = machine.Pin(25,machine.Pin.OUT) while True: LED.value(1) utime.sleep_ms(500) LED.value(0) utime.sleep_ms(500) 10</pre>	~	
	×1	
Consola MMY: SOIT TEDOOT		
Traceback (most recent call last): File " <stdin>", line 7, in <module> KeyboardInterrupt: MPY: soft reboot MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040 Type "help()" for more information. >>></module></stdin>	~	
	MicroPython (Raspberry Pi Pico) · Board	CDC @ COM44

Figura 13. Edición de código en Micropython

Guardar programa.

Tendrá dos opciones a elegir:

- 1. En la computadora.
- 2. En la Raspberry Pi Pico.

Seleccionar: Guardar como.

	Desictop\Raspberry	y 2024/Micropython/GPIO/blink25.py @: 10 : 1	
fichero Editar Visualizar Ejer	iutar Herramienta	is Ayuda	
 Nuevo programa Abrir fichero Archivos recientes 	Ctrl+N Ctrl+O		Asistente
Cerrar Cerrar Guardar Guardar Guardar toriou los ficheros Celerdar como Guardar una copia Mover / renombrar	Ctrl+W Ctrl+Shift+W Ctrl+S Ctrl+S Ctrl+Alt+S Ctrl+Shift+S	chine.Pin.OUT)	^
Imprimir	Ctrl+P		
Salir	Alt+F4		
Consola -			~
Process ended wit	h exit code	None.	^
MicroPython v1.24	.1 on 2024- more infor	11-29; Raspherry Pi Pico with RP2040	
Type "help()" for >>>		100 U.S. 201 L.	
Type "help()" for >>> MPY: soft reboot			1
Type "help()" for >>> MPY: soft reboot MicroPython v1.24	.1 on 2024	-11-29; Raspberry Pi Pico with RP2040	1
Type "help()" for >>> MPY: soft reboot MicroPython v1.24 Type "help()" for	.1 on 2024 more info	-11-29; Raspberry Pi Pico with RP2040 rmation.	1
Type "help()" for >>> MPY: soft reboot MicroPython v1.24 Type "help()" for >>>	.1 on 2024 more info	-11-29; Raspberry Pi Pico with RP2040 rmation.	

Figura 14. Salvar programa actual

- Este computador; lo podrá guardar en alguna ubicación dentro de su computadora con el nombre de su elección NombrePrograma.py; este dejará de correr en cuanto lo detenga o se salga de Thonny (figura 15).
- II. **Salvar en Raspberry Pi Pico**. Permite dejar permanente el programa, debido a que se almacena en memoria de programa ROM (flash), el nombre que debe tener es **main.py**; de manera que es lo primero que ejecutará la Raspberry Pi Pico (figuras 16 y 17).




Thonny - C\Users\ranay\Desktop\Raspberry 2024\Micropython\GPIO\Ejemplo\blink25.py @	9:21	- 🗆 X
Fichero Editar Visualizar Ejecutar Herramientas Ayuda		
B 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		
blink25.py		Asistente
<pre>1 import machine 2 import utime 3 4 LED = machine.Pin(25,machine.Pin.OUT) 5 while True: 6 LED.value(1) 7 utime.sleep_ms(500) 8 LED.value(0)</pre>	The ¿Dónde guardar?	
9 utime.sleep_ms(500)	Este computador Raspberry Pi Pico	
Consola MFI: SOIT REDOOT	^	
<pre>Traceback (most recent call last): File "<stdin>", line 9, in <module> KeyboardInterrupt: MPY: soft reboot MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with Type "help()" for more information. >>></module></stdin></pre>	h RP2040	
	MicroPython (Raspberr	y Pi Pico) • Board CDC @ COM44
■ 🔎 Buscar 🐠 🗧 🖸 🖼	🗧 💫 🎫 💿 💁 🌇 ≷	22°C ^ ^{04:38} p. m. 20/03/2025 💭

Figura 16. Salvar en la Raspberry Pi Pico

	Guardar a Raspberry Pi Pico	×		
blink25.py	Ramberry Di Dice		Asistente	
<pre>import machine import utime LED = machine.Pin(25,mac while True: LED.value(1) LED.value(1) LED.value(0) LED.value(0) utime.sleep_ms(500) </pre>	hine.Pi	Tamaño (bytes)		
	Nombre de archivo: main.py	Ok Cancelar		

Figura 17. Nombrar el código main.py

Ejecutar un programa.

Presionar el icono (Ejecutar el script actual F5) iniciar o (Detener/Reiniciar back–end Ctrl-F2) para suspender la ejecución o reconectar; en caso de haber guardado en la Raspberry Pi Pico, será requerido ejecutar desde Thonny la primera vez que lo descargue, posteriormente al salir del IDE o al conectar a alimentación, el programa se ejecutará de manera automática.

K Thonny - C\Users\ranay\Desktop\Raspberry 2024\Micropython\GPI0\Ejemplo\blink25.py @ 9:21)
Fichero Editar Visualizar Ejecutar Herramientas Ayuda	
18H 0 - XAP 0 =	
<pre>csin nombre> Epv</pre>	Asistente
import utime Detiene ejecución	(A)
4 LED = machi 5 while True: Inicia ejecucion 6 LED.val	
7 utime.sleep_ms(500) 8 LED.value(0) 9 utime.sleep_ms(500)	
Consola	(**)
Traceback (most recent call last): File "main.py", line 7, in <module> KeyboardInterrupt:</module>	· · · · · · · · · · · · · · · · · · ·
MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040 Type "help()" for more information.	
MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040 Type "help()" for more information. MPY: soft reboot	
MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040 Type "help()" for more information. MPY: soft reboot MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040	
MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040 Type "help()" for more information. MPY: soft reboot MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040 Type "help()" for more information.	



En caso de reconectar el cable al entrar a Thonny, e iniciar la ejecución en la Raspberry Pi Pico, o al haber conectado la plataforma con el IDE Thonny ya abierto,

sin tener programa en la plataforma; bastará con presionar el botón 📟 para restablecer la conexión.



Figura 19. Restablecer conexión con Raspberry Pi Pico

Una vez reconectando, la plataforma quedará lista para continuar ejecutando programas nuevos.





Programación de App Inventor Comunicación Bluetooth

Transmisión 'A', D', T', T', S'

APLICACIÓN PARA CONTROLAR DOS MOTORES DE CORRIENTE DIRECTA POR MEDIO DE BLUETOOTH.

OBJETIVO GENERAL: Realizar el software y hardware para comunicar y controlar dos motores de corriente directa, que semejarán la estructura de un robot móvil de tipo diferencial de manera inalámbrica, utilizando Bluetooth.

OBJETIVOS PARTICULARES: Se debe realizar la aplicación APK que correrá en el teléfono celular y transmitirá comandos a través del Bluetooth, estos serán recibidos por otra interfaz Bluetooth, que enviará a Raspberry Pi Pico; recibirá los comandos y generará las acciones pertinentes para generar los movimientos deseados en los motores de cd. Las tablas 1 y dos describen la asignación de señales de control que el microcontrolador generará de acuerdo al comando recibido y las acciones a ejecutar respectivamente.

I. Hardware

El circuito electrónico es el siguiente (figura 1).



Figura 1. Circuito de control de dos motores de CD

	Motor 1			Motor 2	
GPIO0	GPIO1	GPIO20	GPIO2	GPIO3	GPIO19
DIR1_M1	DIR2_M1	EN_M1	DIR1_M2	DIR2_M2	EN_M2

Comando	ACCION		
Puerto serie	MOTOR M1	MOTOR M2	
'S'	PARO	PARO	
'A'	DERECHA	DERECHA	
'T'	IZQUIERDA	IZQUIERDA	
'D'	DERECHA	IZQUIERDA	
"['	IZQUIERDA	DERECHA	

Tabla 1. Asignación de señales de control de los motores de CD

Tabla 2. Control de motores, comunicación serie

II. Aplicación APK

Se ha utilizado el software App Inventor del MIT para construir y programar la aplicación, el diseño de la aplicación se muestra en la figura 2.



Una vez hecha la interfaz, se inicia la programación de los diferentes elementos que la integran, como son los botones, buscar los dispositivos Bluetooth cercanos al teléfono celular, conectarse al Bluetooth deseado, desconectarse y salir de la aplicación.



Figura 3. Programación de la aplicación

Se proporcionará a los alumnos la liga para descargar la aplicación APK para proceder a instalar y ejecutar para iniciar con el control vía Bluetooth, una vez instalada despliega la pantalla de la figura 4.



Figura 4 Aplicación APK

III. Software

El programa que correrá en el microcontrolador será resuelto por el alumno(a), deberá tener como requisito configurar la comunicación a 9600 BAUDS y recibir los valores ASCII de los caracteres A, D, I, T, S en mayúsculas para generar el control.

Raspberry Pi Pico

Control remoto de giro de motor a pasos



Figura 1. dispositivos para implementación del proyecto



Figura 2. Diagrama de bloques del proyecto



Figura 3. el motor de pasos tiene varios métodos para operarlo. En la tabla se detalla cómo deben activarse las fases para obtener resultados distintos en cuanto al torque y el ángulo de giro por paso, dependiendo del método utilizado

Introducción

Raspberry pi pico w integra el protocolo TCP-IP y tiene conectividad Wifi, lo cual permite su conexión a internet a través de un router cercano al módulo, con la posibilidad de control remoto del tipo "Anywhere", desde cualquier parte del mundo. Esto significa que, ya sea a través de un teléfono celular ó una computadora conectados a internet, se pueden enviar comandos al microcontrolador para activar dispositivos y recibir información de sensores ó alarmas instaladas en casas y oficinas.

Objetivos

El proyecto muestra la manera de realizar el control remoto de un motor de pasos 28BYJ-48, desde una PC y con un programa emulador de terminal, con comandos enviados al microcontrolador Raspberry pi pico.

A través de un programa desarrollado en lenguaje Micropython, recibe los comandos y opera un motor de pasos, el cual gira el número de grados indicados por el usuario de la terminal.

Se utilizan herramientas de software gratuitas, como son el firmware Micropython, el software IDE llamado uPyCraft, y el emulador de terminal Termite v3.4. Igualmente los componentes del proyecto.

Descripción

- Con base en las figuras 1 y 2, se muestran los diagramas de conexión del proyecto que utiliza componentes de bajo costo.

- Para desarrollar el proyecto, además del hardware indicado, se requiere primeramente cargar en la PC el driver para la interfaz CH340G. Al conectar la PC con el módulo, por cable MicroUSB, en el administrador de dispositivos deberá darse de alta un nuevo puerto serial COM, que será usado por las herramientas de software posteriormente utilizadas.

- Debe recordarse que el lenguaje Micropython funciona a través de un firmware intérprete que debe ser cargado previamente al microcontrolador. Posteriormente, por medio de un emulador de terminal llamado "Termite v3.4", se envían los comandos al microcontrolador de tal manera de indicar el número de grados de giro del motor, en ambos sentidos (clockwise, counterclockwise).

Siguiendo el diagrama de conexiones mostrado en las figuras 1 y 2, el microcontrolador se conecta con un cable Micro USB a la computadora PC y al motor de pasos por medio de cables dupont. Se observa como el motor se alimenta desde el pin de 3.3v ó de 5v del microcontrolador.
Para la comunicación inicial con el NodeMCU y el almacenamiento del programa de aplicación, se utiliza la herramienta ya mencionada uPyCraft; main.py, es un programa de prueba;

Hace giros del motor en un sentido y el contrario. El programa se carga desde uPyCraft y el funcionamiento es automático ó después del reset. No requiere el uso del emulador Termite. motorGradosSerial-2 éste es el programa de aplicación. Debe primeramente cargarse en el microcontrolador desde uPyCraft. Una vez cargado, se cierra uPycraft y abrimos el emulador de terminal "Termite v3.4". Para escribir el comando para ejecutar el programa, debe copiar y pegar en la línea de comandos de Termite lo siguiente: exec(open('motorGradosSerial-2.py').read(),globals())

```
Código fuente:
from machine import Pin
from time import sleep
pin1 = Pin(16, Pin.OUT)
                           #PINES DE CONEXION DEL MODULO ESP8266 CON EL DRIVER ULN2003
pin2 = Pin(5, Pin.OUT)
pin3 = Pin(4, Pin.OUT)
pin4 = Pin(0, Pin.OUT)
tiempo = 0.005
#tiempo = 0.5
WaveDrive=[
  [1,0,0,0],
  [0, 1, 0, 0],
  [0, 0, 1, 0],
  [0, 0, 0, 1]]
                           #SECUENCIA DE ONDA
FullStepDrive=[
  [1,0,0,1],
  [1,1,0,0],
  [0, 1, 1, 0],
  [0, 0, 1, 1]]
                             #SECUENCIA DE PASO COMPLETO
```

```
HalfStepDrive=[
  [1, 0, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 0],
  [0,1,1,0],
  [0, 0, 1, 0],
  [0, 0, 1, 1],
  [0, 0, 0, 1]]
                          #SECUENCIA DE MEDIO PASO
#Secuencias sentido de giro HORARIO
def secuenciaWaveDrive horario(tiempo):
  for indice in range(2048):
   pin1.value(WaveDrive[indice%4][3])
   pin2.value(WaveDrive[indice%4][2])
   pin3.value(WaveDrive[indice%4][1])
   pin4.value(WaveDrive[indice%4][0])
   sleep(tiempo)
def secuenciaFullStepDrive horario(tiempo):
  for indice in range(2048):
   pin1.value(FullStepDrive[indice%4][3])
   pin2.value(FullStepDrive[indice%4][2])
   pin3.value(FullStepDrive[indice%4][1])
   pin4.value(FullStepDrive[indice%4][0])
   sleep(tiempo)
def secuenciaHalfStepDrive horario(tiempo):
  for indice in range(4096):
   pin1.value(HalfStepDrive[indice%8][3])
   pin2.value(HalfStepDrive[indice%8][2])
   pin3.value(HalfStepDrive[indice%8][1])
   pin4.value(HalfStepDrive[indice%8][0])
   sleep(tiempo)
#Secuencias sentido de giro AHORARIO
def secuenciaWaveDrive Ahorario(tiempo):
  for indice in range(2048):
   pin1.value(WaveDrive[indice%4][0])
   pin2.value(WaveDrive[indice%4][1])
   pin3.value(WaveDrive[indice%4][2])
   pin4.value(WaveDrive[indice%4][3])
   sleep(tiempo)
def secuenciaFullStepDrive Ahorario(tiempo):
  for indice in range(2048):
   pin1.value(FullStepDrive[indice%4][0])
   pin2.value(FullStepDrive[indice%4][1])
   pin3.value(FullStepDrive[indice%4][2])
   pin4.value(FullStepDrive[indice%4][3])
   sleep(tiempo)
def secuenciaHalfStepDrive Ahorario(tiempo):
  for indice in range(4096):
   pin1.value(HalfStepDrive[indice%8][0])
```

```
pin2.value(HalfStepDrive[indice%8][1])
  pin3.value(HalfStepDrive[indice%8][2])
  pin4.value(HalfStepDrive[indice%8][3])
  sleep(tiempo)
def secuenciaParo():
 pin1.value(0)
 pin2.value(0)
 pin3.value(0)
 pin4.value(0)
 sleep(tiempo)
def main():
 secuenciaWaveDrive horario(tiempo)
 secuenciaWaveDrive Ahorario(tiempo) #GIRO COMPLETO
 secuenciaFullStepDrive horario(tiempo)
 secuenciaFullStepDrive Ahorario(tiempo) #GIRO COMPLETO
 secuenciaHalfStepDrive horario(tiempo/2)
 secuenciaHalfStepDrive Ahorario(tiempo/2) #GIRO COMPLETO
 secuenciaParo()
main()
```

Raspberry Pi Pico

Control manual de posición angular en un motor a pasos, mediante un potenciómetro de precisión



Figura 1. Diagrama de conexiones del proyecto. Incluye un microcontrolador, un motor de pasos BYJ-48, un display OLED SSD1306 y un potenciómetro de precisión para el control manual.



Figura 2. detalle del display OLED SSD1306, de 128 x 64 pixeles, 0.96 pulgadas

Descripción general

En este proyecto se realiza el control en forma manual y por medio de un potenciómetro de precisión. Adicionalmente, se cuenta con un display OLED de alta definición para mostrar los datos del valor deseado y valor final del ángulo de giro del motor, ahorrando la necesidad del uso de un emulador de terminal, ya que todas las componentes del proyecto están conectadas en un espacio reducido.

Se hace uso de la función "Autorun" del lenguaje Micropython.

Con ayuda de las figuras 1 y 2, se hacen las conexiones de los componentes usando cables del tipo dupont, como se muestra.

Para hacer el programa Autorun, se carga el archivo llamado main.py. Una vez cargado, puede prescindir de la computadora y conectar directamente al microcontrolador. El programa se ejecuta automáticamente al encender la fuente.

Código fuente, main.py

```
from machine import Pin, ADC, I2C
import ssd1306
from time import sleep
i2c = I2C(scl=Pin(5), sda=Pin(4))
pot=ADC(0)
```

```
oled width = 128
oled height = 64
oled = ssd1306.SSD1306 I2C(oled width, oled height, i2c)
pin1 = Pin(15, Pin.OUT)
pin2 = Pin(13, Pin.OUT)
pin3 = Pin(12, Pin.OUT)
pin4 = Pin(14, Pin.OUT)
tiempo = 0.005
WaveDrive=[
 [1,0,0,0],
  [0, 1, 0, 0],
  [0, 0, 1, 0],
 [0, 0, 0, 1]
FullStepDrive=[
  [1, 0, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 1, 0],
 [0, 0, 1, 1]]
HalfStepDrive=[
  [1, 0, 0, 1],
 [1,0,0,0],
 [1, 1, 0, 0],
  [0,1,0,0],
  [0,1,1,0],
  [0, 0, 1, 0],
  [0, 0, 1, 1],
  [0, 0, 0, 1]
#Secuencias sentido de giro HORARIO wavedrive
def secuenciaWaveDrive horario(tiempo):
 for indice in range(2048):
   pin1.value(WaveDrive[indice%4][3])
   pin2.value(WaveDrive[indice%4][2])
   pin3.value(WaveDrive[indice%4][1])
   pin4.value(WaveDrive[indice%4][0])
   sleep(tiempo)
#Secuencias sentido de giro AHORARIO wavedrive
def secuenciaWaveDrive Ahorario(tiempo):
 for indice in range(2048):
   pin1.value(WaveDrive[indice%4][0])
   pin2.value(WaveDrive[indice%4][1])
   pin3.value(WaveDrive[indice%4][2])
   pin4.value(WaveDrive[indice%4][3])
   sleep(tiempo)
#Secuencia de paro
def secuenciaParo():
 pin1.value(0)
```

```
pin2.value(0)
 pin3.value(0)
 pin4.value(0)
 sleep(tiempo)
#Secuencia gira n grados, horario, antihorario
#_____
def giroGrados(grados, tiempo):
 if(grados>=0):
   for indice in range(int((2048/360)*grados)):
     pin1.value(WaveDrive[indice%4][0])
                                                    #horario
     pin2.value(WaveDrive[indice%4][1])
     pin3.value(WaveDrive[indice%4][2])
     pin4.value(WaveDrive[indice%4][3])
     sleep(tiempo)
 else:
   for indice in range(int((2048/360)*abs(grados))):
     pin1.value(WaveDrive[indice%4][3])
                                                    #Ahorario
     pin2.value(WaveDrive[indice%4][2])
     pin3.value(WaveDrive[indice%4][1])
     pin4.value(WaveDrive[indice%4][0])
     sleep(tiempo)
#Funcion principal
#_____
def main():
 i =0
 pos actual=0
          #0123456789012345
 oled.text(' Control de giro', 0, 0)
 oled.text('angular mediante', 0, 15)
 oled.text(' potenciometro ', 0, 30)
 oled.text('DESPLAZANDO....', 0, 45)
 oled.show()
 while(True):
   nueva posicion=2*pot.read()
   if(nueva posicion>pos actual):
     pos actual=pos actual+1
     pin1.value(WaveDrive[pos actual%4][3])
                                           #Horario
     pin2.value(WaveDrive[pos_actual%4][2])
     pin3.value(WaveDrive[pos_actual%4][1])
     pin4.value(WaveDrive[pos_actual%4][0])
     sleep(tiempo)
   elif(nueva posicion<pos actual):</pre>
     pos actual=pos actual-1
     pin1.value(WaveDrive[pos actual%4][3])
                                          #Ahorario
     pin2.value(WaveDrive[pos actual%4][2])
     pin3.value(WaveDrive[pos actual%4][1])
     pin4.value(WaveDrive[pos actual%4][0])
     sleep(tiempo)
   else:
     pin1.value(0)
                                           #Paro
     pin2.value(0)
     pin3.value(0)
     pin4.value(0)
     sleep(tiempo)
```

```
oled.fill(0)
     oled.text('Desplazamiento:', 0, 0)
     oled.text(<mark>'Ajuste: ' , 0, 22</mark>)
     oled.text('Posicion: ', 0, 38)
     oled.text(grado ajuste, 60, 22)
     oled.text(grado actual, 74, 38)
     oled.show()
   i=i+1
   if(i>50):
     grado actual=str(int((360/2048)*pos actual))
     grado ajuste=str(int((360/2048)*nueva posicion))
     print('Posicion actual: '+grado actual+' grados, Posicion de ajuste:
'+grado_ajuste+' grados')
     #oled.text(grado ajuste, 74, 22)
     #oled.text(grado actual, 74, 38)
     #oled.show()
     i=0
#Programa principal
#_____
main()
#_____
from machine import Pin, ADC, I2C
import ssd1306
from time import sleep
i2c = I2C(scl=Pin(5), sda=Pin(4))
pot=ADC(0)
oled width = 128
oled height = 64
oled = ssd1306.SSD1306 I2C(oled width, oled height, i2c)
pin1 = Pin(15, Pin.OUT)
pin2 = Pin(13, Pin.OUT)
pin3 = Pin(12, Pin.OUT)
pin4 = Pin(14, Pin.OUT)
tiempo = 0.005
WaveDrive=[
  [1,0,0,0],
  [0, 1, 0, 0],
  [0,0,1,0],
  [0, 0, 0, 1]
FullStepDrive=[
  [1,0,0,1],
  [1,1,0,0],
  [0, 1, 1, 0],
```

```
[0,0,1,1]]
```

```
HalfStepDrive=[
 [1,0,0,1],
 [1, 0, 0, 0],
 [1,1,0,0],
 [0,1,0,0],
 [0,1,1,0],
 [0,0,1,0],
 [0, 0, 1, 1],
 [0, 0, 0, 1]
#Secuencias sentido de giro HORARIO wavedrive
#_____
def secuenciaWaveDrive horario(tiempo):
 for indice in range(2048):
   pin1.value(WaveDrive[indice%4][3])
   pin2.value(WaveDrive[indice%4][2])
   pin3.value(WaveDrive[indice%4][1])
   pin4.value(WaveDrive[indice%4][0])
   sleep(tiempo)
#Secuencias sentido de giro AHORARIO wavedrive
#_____
def secuenciaWaveDrive Ahorario(tiempo):
 for indice in range(2048):
   pin1.value(WaveDrive[indice%4][0])
   pin2.value(WaveDrive[indice%4][1])
   pin3.value(WaveDrive[indice%4][2])
   pin4.value(WaveDrive[indice%4][3])
   sleep(tiempo)
#Secuencia de paro
#_____
def secuenciaParo():
 pin1.value(0)
 pin2.value(0)
 pin3.value(0)
 pin4.value(0)
 sleep(tiempo)
#Secuencia gira n grados, horario, antihorario
def giroGrados(grados, tiempo):
 if(grados>=0):
   for indice in range(int((2048/360)*grados)):
    pin1.value(WaveDrive[indice%4][0])
                                             #horario
     pin2.value(WaveDrive[indice%4][1])
    pin3.value(WaveDrive[indice%4][2])
    pin4.value(WaveDrive[indice%4][3])
    sleep(tiempo)
 else:
   for indice in range(int((2048/360)*abs(grados))):
    pin1.value(WaveDrive[indice%4][3])
                                              #Ahorario
    pin2.value(WaveDrive[indice%4][2])
     pin3.value(WaveDrive[indice%4][1])
     pin4.value(WaveDrive[indice%4][0])
     sleep(tiempo)
```

```
#_____
def main():
 i=0
 pos actual=0
           #0123456789012345
 oled.text(' Control de giro', 0, 0 )
 oled.text('angular mediante', 0, 15)
 oled.text(' potenciometro ', 0, 30)
 oled.text(' DESPLAZANDO....', 0, 45)
 oled.show()
 while(True):
   nueva posicion=2*pot.read()
   if(nueva posicion>pos actual):
     pos actual=pos actual+1
     pin1.value(WaveDrive[pos actual%4][3])
                                             #Horario
     pin2.value(WaveDrive[pos actual%4][2])
     pin3.value(WaveDrive[pos actual%4][1])
     pin4.value(WaveDrive[pos actual%4][0])
     sleep(tiempo)
   elif(nueva posicion<pos actual):</pre>
     pos actual=pos actual-1
     pin1.value(WaveDrive[pos actual%4][3])
                                             #Ahorario
     pin2.value(WaveDrive[pos_actual%4][2])
     pin3.value(WaveDrive[pos actual%4][1])
     pin4.value(WaveDrive[pos actual%4][0])
     sleep(tiempo)
   else:
     pin1.value(0)
                                             #Paro
     pin2.value(0)
     pin3.value(0)
     pin4.value(0)
     sleep(tiempo)
     oled.fill(0)
     oled.text('Desplazamiento:', 0, 0)
     oled.text('Ajuste: ' , 0, 22)
     oled.text('Posicion: ', 0, 38)
     oled.text(grado_ajuste, 60, 22)
     oled.text(grado_actual, 74, 38)
     oled.show()
   i=i+1
   if(i>50):
     grado actual=str(int((360/2048)*pos actual))
     grado ajuste=str(int((360/2048)*nueva posicion))
     print('Posicion actual: '+grado actual+' grados, Posicion de ajuste:
'+grado ajuste+' grados')
     #oled.text(grado ajuste, 74, 22)
     #oled.text(grado actual, 74, 38)
     #oled.show()
```

```
i=0
```

#Funcion principal

main()

Raspberry Pi Pico 2 W Giroscopio y acelerómetro

RESUMEN

En este documento se intenta crear una guía para la utilización del módulo MPU-6050, que consiste en un acelerómetro y un giroscopio, de 3 ejes, utilizando una tarjeta raspberry pi pico 2 W.

En esta misma guía se abordarán rápidamente las características del módulo mencionado, el software que se utilizará para configurar la comunicación entre dispositivos y programar alguna función en la Raspberry.

MATERIALES

- Raspberry pi pico 2 W



- Módulo MPU-6050



- Headers para colocar o soldar en la raspberry
- Jumpers o cables de conexión para los headers

DESCRIPCIÓN Y CARACTERÍSTICAS DEL MODULO MPU-6050

Este módulo es un sensor de movimiento, está diseñado para funcionar en tarjetas de desarrollo como Arduino y Raspberry, consta de un acelerómetro y un giroscopio, cada uno de ellos con 3 ejes, económico y de bajo consumo de energía. Debido a su alta sensibilidad, es utilizado para diferentes aplicaciones, por ejemplo, robótica, celulares o dispositivos de videojuegos.

Consta de los siguientes componentes, características y funciones:

- Giroscopio de 3 ejes, con convertidor analógico-digital de 16 bits, programable para los rangos: +/-250, +/-500, +/-1000, +/-2000 dps (grados por segundo)
- Acelerómetro de 3 ejes, con convertidor analógico-digital de 16 bits, programable para los rangos: +/-2g, +/-4g, +/-8g, +/-16g
- Interfaces de comunicación I2C y SPI
- El módulo tiene acceso a 8 pines, de los 24 que tiene en su totalidad el sensor.
- Rango de voltaje de alimentación: 2.375V a 3,46V
- Consumo de corriente: 3.9 mA en modo normal, 5 uA en modo de reposo
- Rango de temperatura del sensor: -40 a 85 grados centígrados.

Pines del módulo:

- Vcc: Pin de alimentación del módulo, acepta entre 3.3V y 5V, dependiendo del regulador.
- GND: Pin de tierra.
- SCL: Línea de reloj I2C, usado para la comunicación con el microcontrolador
- SDA: Línea de datos I2C, usado para la comunicación con el microcontrolador
- XDA: Línea de datos I2C auxiliar, para conectar sensores externos I2C
- XCL: Línea de reloj I2C auxiliar, para conectar sensores externos I2C
- AD0: Pin de selección de dirección I2C esclavo. Utilizado cuando se tiene más de un módulo MPU-6050
- INT Pin de interrupción, utilizado para indicar al microcontrolador que un dato está disponible para ser leído desde un módulo MPU-6050

CONEXIÓN CON RASPBERRY PI PICO

Para este paso sólo se requieren los pines Vcc, GND, SCL y SDA del módulo MPU-6050.

La raspberry pi pico 2 W tiene 2 controladores I2C (I2C0 e I2C1), se puede configurar cada uno de ellos, a través de diferentes pines GPIO (general purpose input output) como se observa en la imagen, pero es importante asignar por código los pines que se utilizarán



En este caso se usará el I2C0 con los pines GP0 y GP1, como se muestra en la siguiente imagen, ya que la funcionalidad de esos pines coincide con los de una Raspberry pi pico.



SOFTWARE, CÓDIGO Y LIBRERÍAS PARA UTILIZAR EL MPU-6050

La interfaz de desarrollo propuesta es Thonny, que es una IDE basada en Python para microcontroladores. Se puede descargar en la siguiente página <u>https://thonny.org/</u>.

Por otro lado, se cargará el software MicroPython, que es un lenguaje de programación que se puede ejecutar en la Rapsberry, se puede descargar desde aquí <u>https://www.raspberrypi.com/documentation/microcontrollers/micropython.html</u> en este caso en particular, usamos el apropiado para Raspberry pi pico 2 W.

Para cargar MicroPython en la Raspberry primero conectamos la Raspberry a la computadora y apretar el botón bootsel, se abrirá una carpeta de la memoria de la Raspberry, ahí copiamos el archivo descargado desde la página anterior.

Otra manera de descargar el archivo es ejecutar Thonny y en el menú Ejecutar>Configurar intérprete>Intérprete encontraremos una opción "Instala o actualiza MicroPython", seleccionamos la familia RP2, variante Raspberry Pi Pico 2 W, nos dará opción de elegir la variante correspondiente y finalmente aceptamos instalar.

Después de esos procesos se reiniciará la Raspberry y es necesario reiniciar Thonny para ejecutar el IDE de manera correcta.

Tha Thonny - <sin nombre=""></sin>	@ 1:1	×
Fichero Editar Visualizar Eje	cutar Herramientas Ayuda	
0 😸 🖬 🛛 0 🔹 🧐 3e 🛛		
Archivos -	<sin nombre=""> ×</sin>	
Este computador = ^ C: \ Users \ isryo	1	^
 3D Objects ansel Contacts curso java Desktop Documents Favorites Links 		>
🖽 🖗 Music	Consola	
Raspberry Pi Pico =	<pre>MicroPython v1.25.0-preview.393.gf1 018ee5c on 2025-03-17; Raspberry Pi Pico 2 W with RP2350 Type "help()" for more information. >>></pre>	^
~		\sim
	MicroPython (Raspberry Pi Pico) • Board CDC @ COM3	3 ≡

En la sección consola nos indica sobre qué tarjeta se está programando.

Se utilizarán las dos siguientes librerías para la configuración de los componentes del MPU-6050 que ejecutará nuestra raspberry: <u>imu.py</u> y <u>vector3d.py</u>, las cuales pueden ser descargadas desde el siguiente enlace <u>https://github.com/micropython-imu/micropython-mpu9x50</u>.

El código que podemos utilizar es el siguiente y lo guardaremos con el nombre main.py:

from imu import MPU6050 from time import sleep from machine import Pin, I2C

```
i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)
imu = MPU6050(i2c)
```

```
while True:
    ax=round(imu.accel.x,2)
    ay=round(imu.accel.y,2)
    az=round(imu.accel.z,2)
    gx=round(imu.gyro.x)
    gy=round(imu.gyro.y)
    gz=round(imu.gyro.z)
    tem=round(imu.temperature,2)
    print("ax",ax,"\t","ay",ay,"\t","az",az,"\t","gx",gx,"\t","gy",gy,"\t","gz",gz,"\t","Temperature",tem,"
    ",end="\r")
    sleep(0.2)
```

CARGANDO LOS CÓDIGOS

Utilizaremos Thonny para subir las librerías y el código a la Raspberry. Para ello iremos a menú Visualizar>archivos (debe estar activado) y en el lado izquierdo buscaremos la carpeta donde guardamos cada uno de nuestros códigos y los cargaremos en la Raspberry, dando click derecho sobre el nombre del archivo y enseguida aplicamos subir a/ como se muestra en la siguiente imagen.

Archivos	main.py
Este computador = ^ C:\Users\isryo	<pre>1 from imu import MPU6050 2 from time import sleep 3 from machine import Pin, I2C</pre>
 ♥ imu.py ♥ main.py A P mpu6050.jp ♥ mpu6050.p m po 600000000000000000000000000000000000	prir en Thonny prir en la aplicación externa por defecto onfigurando .py archivos ostrar archivos ocultos
Contraction of the second seco	bira/ ccel.x,2) cchivo nuevo ccel.y,2) uevo Directorio ccel.y,2) ortar ccel.z,2)
Raspberry Pi Pico P R R N	oppiar yro.x) agar yro.y) anombrar yro.z) over al basurero temperature,2)
Р	opiedades sieep(0.2)

Una vez que se subieron correctamente los códigos aparecerán los mismos en la sección de la memoria de la raspberry utilizada.



Una vez terminado todo este proceso se puede ejecutar el proyecto y observar los resultados en la consola.

Raspberry Pi Pico

Control de motor de Corriente Directa

Los motores de DC son ampliamente utilizados en diferentes proyectos de robótica y electrónica. Dada la versatilidad de la Raspberry Pi Pico, su tamaño y los diferentes sensores y actuadores que se pueden conectar, además del bajo consumo de energía es una opción muy importante a considerar, utilizar este microcontrolador para el manejo de este tipo de motores.

L293D

Este componente es un puente H de 4 canales. La energía que puede proporcionar al motor está en un rango de 4.6-36 V y una corriente de hasta 600 mA, para que la lógica en el componente sea funcional, requiere 5V de alimentación.

En la imagen se observan los habilitadores de los canales (EN), las entradas lógicas (A) y las salidas de potencia que van directamente al motor (Y), los pines de tierra (GROUND) van conectados a una tierra común, Vcc1 es la alimentación lógica del circuito (5V) y Vcc2 es la alimentación de voltaje con el que opera el motor de DC.

CONEXIÓN AL MOTOR DE DC



PROCEDIMIENTO PARA EL MANEJO DEL MOTOR

Como se puede ver en el diagrama propuesto, se utilizan los pines de propósito general como salidas. Para crear el código se utiliza la IDE Thonny, que se mencionó en el proyecto propuesto anteriormente. Con el siguiente código aplicado a las conexiones de la imagen anterior, que es compatible con el modelo de Raspberry Pi Pico.

from machine import Pin, PWM from time import sleep

```
pwmPIN=16
cwPin=14
acwPin=15
def motorMove(speed,direction,speedGP,cwGP,acwGP):
       if speed > 100: speed=100
       if speed < 0: speed=0
       Speed = PWM(Pin(speedGP))
       Speed.freq(50)
       cw = Pin(cwGP, Pin.OUT)
       acw = Pin(acwGP, Pin.OUT)
       Speed.duty u16(int(speed/100*65536))
       if direction < 0:
          cw.value(0)
          acw.value(1)
       if direction == 0:
         cw.value(0)
          acw.value(0)
       if direction > 0:
         cw.value(1)
          acw.value(0)
```

main program motorMove(100,1,pwmPIN,cwPin,acwPin) sleep(5) motorMove(100,0,pwmPIN,cwPin,acwPin)

Como se observa en el código, el PIN 16 se utiliza como salida PWM, que va conectada directamente al habilitador de los canales 1 y 2 del circuito L293D.

También se observa que hay 5 parámetros a considerar en el código.

El primero es la velocidad que va entre 0 y 100, el cual indica el ciclo de trabajo en el PWM.

El segundo es la dirección de la rotación del motor, los valores positivos crean un giro en sentido horario, mientras que los valores negativos crean un sentido de giro antihorario, ambos deben ser de tipo entero.

El tercer parámetro envía el valor de PWM, a través del pin utilizado como salida PWM, dicha salida define la velocidad del motor.

El cuarto y quinto parámetro son salidas digitales que se activan o desactivan dependiendo del valor que se haya colocado en el parámetro "dirección"

El código propuesto se guarda y se carga directamente en la Raspberry, de la misma manera que se carga el archivo main.py del proyecto propuesto anteriormente.

Referencias

https://thonny.org/

https://www.raspberrypi.com/

https://micropython.org/

https://cpulator.01xz.net/

https://www.raspberrypi.com/documentation/

https://developer.arm.com/documentation