AlbertaLoop Mechatronics
Showdown 2025

Introduction to the Raspberry Pi Pico

Steven Knudsen, PhD PEng
knud@ualberta.ca

Outline

1. Raspberry Pi Pico and Pico W overview
2. Setting up for MicroPython
3. Setting up for C/C++
4. What about Rust?
5. Wokwi.com and Pico

Sources for the talk

Coming soon…

github.com/StevenKnudsen/AlbertaLoop2025Workshop

Assumptions

- You know how to program in Python
- You have a Pico

Raspberry Pi Pico and Pico W

Raspberry Pi Pico is a low-cost, high-performance microcontroller board with flexible digital interfaces. Key features include:

- RP2040 microcontroller chip designed by Raspberry Pi in the United Kingdom
- Dual-core Arm Cortex M0+ processor, flexible clock running up to 133 MHz
- 264 kB of SRAM, and 2MB of on-board flash memory
- USB 1.1 with device and host support
- Low-power sleep and dormant modes
- Drag-and-drop programming using mass storage over USB
- 26 × multi-function GPIO pins
- 2 × SPI, 2 × I2C, 2 × UART, 3 × 12-bit ADC, 16 × controllable PWM channels
- Accurate clock and timer on-chip
- Temperature sensor
- Accelerated floating-point libraries on-chip
- 8 × Programmable I/O (PIO) state machines for custom peripheral support

Images from digikey.ca

Raspberry Pi Pico 2 and Pico 2 W

Raspberry Pi Pico 2 updates the Pico with:

- Dual Arm Cortex-M33 or dual Hazard3 RISC-V processors @ 150MHz
- 520 KB on-chip SRAM
- 2.4GHz 802.11n wireless LAN and Bluetooth 5.2 (Raspberry Pi Pico 2 W only)
- Software- and hardware-compatible with Raspberry Pi Pico 1
- Robust and fully documented security features:
    - ARM TrustZone for Cortex-M
    - Optional boot signing, enforced by on-chip mask ROM, with key fingerprint in OTP
    - Protected OTP storage for optional boot decryption key
    - Global bus filtering based on Arm or RISC-V security/privilege levels
    - Peripherals, GPIOs, and DMA channels individually assignable to security domains
    - Hardware mitigations for fault injection attacks
    - Hardware SHA-256 accelerator
- 2 × SPI, 2 × I2C, 2 × UART, 24 × PWM, 3 x 12-bit ADC
- 1 × USB 1.1 controller and PHY, with host and device support
- 12 × PIO state machines
- Open source C/C++ SDK, MicroPython support

Images from digikey.ca and PiShop.ca

Pico & Pico W – Pico 2 & Pico 2 W

$CA6.10 & $CA9.15

$CA7.63 & $CA10.68

at mouser.ca

Pico will be core for Mechatronics

Getting started with the Pico - MicroPython

The main reference is a very good tutorial

https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/0

Pico on breadboard

https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/1

RPi Pico Pins

XIAO RP2040 and Expansion board

XIAO RP2040 Pins

Install Thonny (Win/Lin/macOS)

Pico in boot select (bootsel) mode

Unplug Pico from USB

Press and hold bootsel button

Plug Pico into USB and wait a few seconds

Release bootsel button

Open Thonny

https://wiki.seeedstudio.com/XIAO-RP2040/
https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/3

Install MicroPython

Install dialogue

Connect

Thonny - <untitled> @ 1 : 1
<untitled>
Assistant
Shell
MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
Local Python 3 • Thonny's Python
MicroPython (Raspberry Pi Pico) • Board in FS mode @ /dev/cu.usbmodem2101
MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem2101
Configure interpreter...
MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem2101

Thonny - <untitled> @ 1 : 1
<untitled>
Assistant
Shell
MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem2101

Success!

Thonny - <untitled> @ 1 : 1
<untitled>
Assistant
Shell
MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> print("Hello AlbertaLoop!")
Hello AlbertaLoop!
>>>
MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem2101

GPIOs

Lots of support for GPIO and other peripherals at MicroPython.org

Quick Reference for MicroPython is found at

https://docs.micropython.org/en/latest/rp2/quickref.html

Support for

- HW Timers, GPIO, UART, PWM, ADC, SW SPI, HW SPI, SW I2C, HW I2C, I2S, RTC, WDT, OneWire, NeoPixel, APA106
- … but probably not …

AI ASSISTANT

MONKEYUSER.COM

monkeyuser.com

Simple RGB blinky

NeoPixel

NEO_PWR
RGB1 LED_RGB_4P
4
VCC
DOUT
1
3
DIN
VSS
2
NEOPIX
12
13
14
GPIO9
GPIO10
GPIO11
GPIO12
GPIO13
GPIO14
GPIO15
TESTEN
15
16
17
18
19
NEOPIX

Load NeoPixel Library

`ws2812.py`

Save Library

Save As to the RP2040

NeoPixel code

Save As to computer or RP2040

Then run

```
from ws2812 import WS2812
import utime
import machine
power = machine.Pin(11,machine.Pin.OUT)
power.value(1)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
YELLOW = (255, 150, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)
BLUE = (0, 0, 255)
PURPLE = (180, 0, 255)
WHITE = (255, 255, 255)
COLORS = (BLACK, RED, YELLOW, GREEN, CYAN, BLUE, PURPLE, WHITE)

led = WS2812(12,1)#WS2812(pin_num,led_count)

while True:
    print("Beautiful color")
    for color in COLORS:
        led.pixels_fill(color)
        led.pixels_show()
        utime.sleep(0.2)
```

Load OLED Library

`ssd1306.py`

Save As to the RP2040

OLED code

Save As to computer or RP2040

Then run

```
from ssd1306 import SSD1306_I2C
from machine import Pin, I2C
from time import sleep

i2c = I2C(1, scl=Pin(7), sda=Pin(6), freq=200000)#Grove - OLED Display 0.96
oled = SSD1306_I2C(128, 64, i2c)

while True:
    oled.fill(0)#clear
    oled.text("Hello, ",0,0)
    oled.text("AlbertaLoop!",0,12)
    oled.show()
    #sleep(0.5)
```

Analogue to Digital Conversion

XIAO RP2040 Pins

XIAO RP2040 Pins

ADC code

Save As to computer
or RP2040

Then run

```
from machine import ADC

adc = ADC(26)        # create an ADC object acting on a pin

while True:
    val = adc.read_u16()  # read a raw analog value in the range 0-65535
    print(val)
```

WOKWi
Simulate IoT Projects in Your Browser
Discord Community
LinkedIn Group
Simulate with Wokwi Online
uno
ARDUINO
Arduino (Uno, Mega, Nano)
ESP32
ESP32
STM32
STM32
Pi Pico

What is Wokwi.com? (from docs.wokwi.com)

Wokwi is an online Electronics simulator. You can use it to simulate Arduino, ESP32, STM32, and many other popular boards, parts and sensors.

Here are some quick examples of things you can make with Wokwi:

- Arduino Uno "Hello World"
- Blink an LED on ESP32
- Monitor the weather on ATtiny85
- Control 32 Servos with Arduino Mega
- Animate an LED Matrix with FastLED
- 7 Segment Counter with MicroPython on ESP32

...and Pico!

Raspberry Pi Pico Simulator

A faster way to prototype Pi Pico projects

Featured projects

Traffic Light

LCD 1602

Speaker Buzzer

Stepper Motor

Mini Piano

7 Segment Counter

KeyPad and LEDs

Simple RGB blinky

Thonny - /Users/knud/Downloads/HackEDbetaPico01.py @ 12 : 40

HackEDbetaPico01.py

```python
import rp2
import time

from machine import Pin

p0 = Pin(0, Pin.OUT)
p1 = Pin(1, Pin.OUT)
p2 = Pin(2, Pin.OUT)

while True :
    p0.off() # pull low to turn on LED
    p1.on()  # pull high to turn off LED
    p2.on()  # pull high to turn off LED
    time.sleep(0.25)
    p0.on()  # pull high to turn off LED
    p1.off() # pull low to turn on LED
    p2.on()  # pull high to turn off LED
    time.sleep(0.25)
    p0.on()  # pull high to turn off LED
    p1.on()  # pull high to turn off LED
    p2.off() # pull low to turn on LED
    time.sleep(0.25)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT

  MPY: soft reboot
```

MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem1324101

Wokwi Version

Trivia

...with wee prizes...

Trivia

What is an equivalent to the U of A’s main frame computer from the 80s?

Amdahl 470 v/6

Date Introduced : 1975

Dimensions overall: 63” x 70” x 26”

Keywords : Clones; Plug compat; IBM

Speed : 3.5 MIPS

Memory Size : up to 8MB

Memory Width : 32-bit

Cost : $3,750,000 (2020 $17,981,250)

Arduino

Date Introduced : 2010

Dimensions overall: 2.7” x 2.1” x 0.6”

Speed : 8 - 11 MIPS

Memory Size : 32k FLASH 2k SRAM

Cost : ~$20

Cost to make : < $5

Trivia

What are the most loved and hated languages in the 2022 Dice survey

Most loved

Rust 86.83%

Elixir 75.46%

Clojure 75.23%

TypeScript 73.46%

Julia 72.51%

Python 67.34%

Most hated

VBA (78.56 percent).

COBOL (hated by 79.96 percent of developers)

MATLAB (80.84 percent), and

Most hated

VBA (78.56 percent).

COBOL (hated by 79.96 percent of developers)

MATLAB (80.84 percent), and

STACK OVERFLOW
YOU ARE BY FAR THE WORST
PROGRAMMING LANGUAGE I HAVE EVER HEARD OF
VISUAL BASIC
BUT YOU HAVE HEARD OF ME
imgflip.com