Embedded systems programming

Sparrow 2024

Florian Topeza
SCALAR 6 Embedded systems department

November 28, 2024

Contents

- Introduction to microcontrollers
- Overview of the Raspberry Pi Pico
- Pinout

- Thonny IDE and MicroPython
- Libraries
- Classes
- Writing data on your microcontroller

- Space embedded systems *by Merlin Kooshmanian*

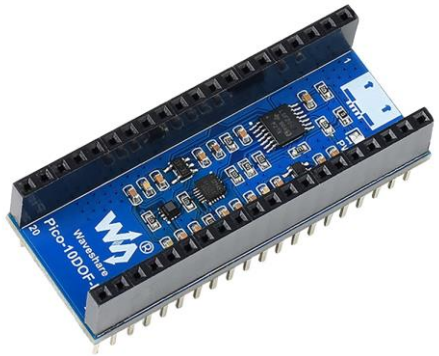
What are the embedded systems in a Sparrow rocket ?

- Buzzer

- Actuator

- Sensors (IMU & barometer)

How to control them ?



What to use ?

- Microcomputer ?
- Microcontroller ?
- Which one ?

We'll use the Raspberry Pi Pico microcontroller.

Overview of the Raspberry Pi Pico

Raspberry Pi Pico

Processor

RP2040, 2 cores, 133MHz

ATPMega 328, 1 core, 16MHz

Flash Memory

2MB

32KB

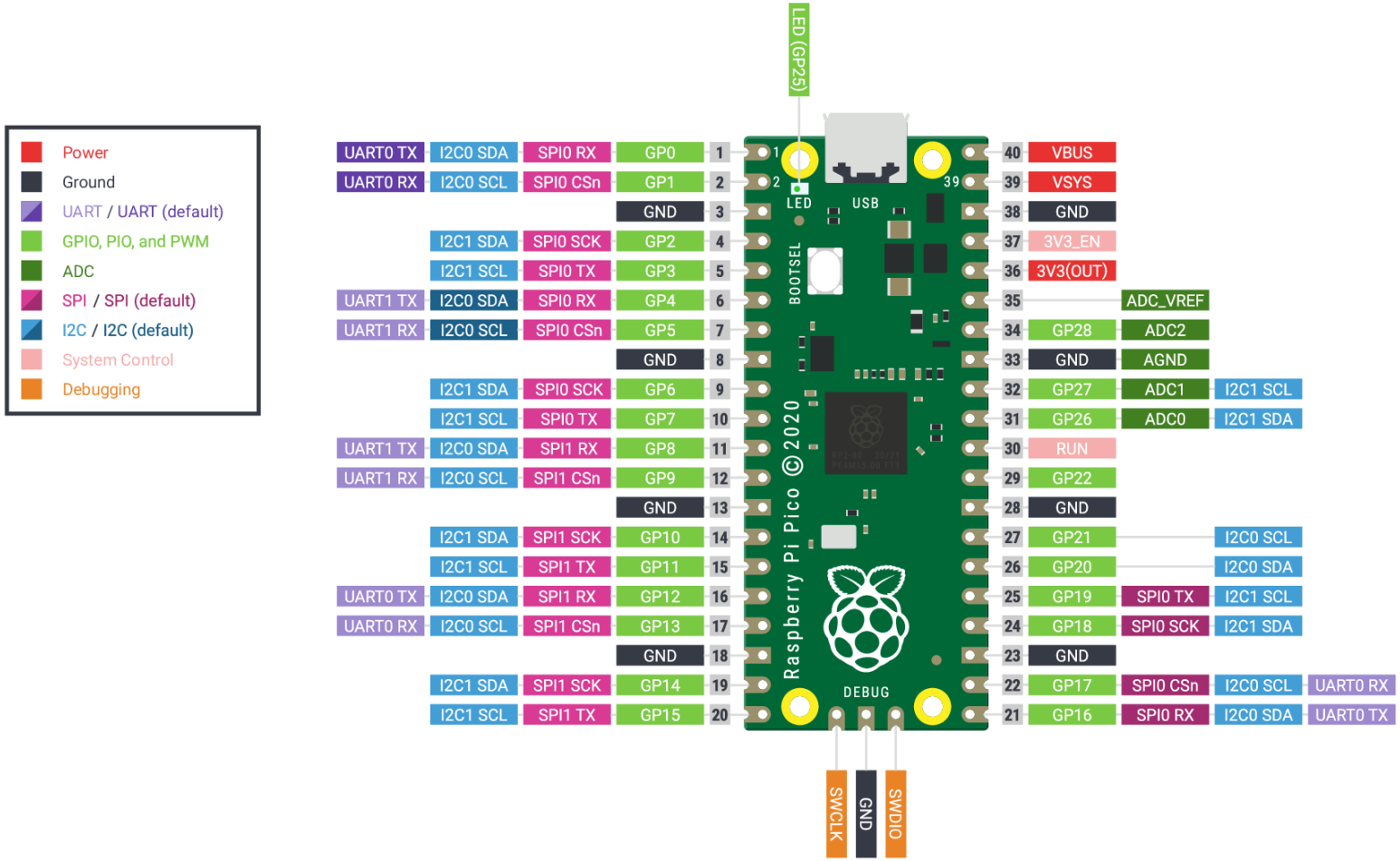
Cost

4 USD

20USD

Arduino Nano

Raspberry Pi Pico Pinout

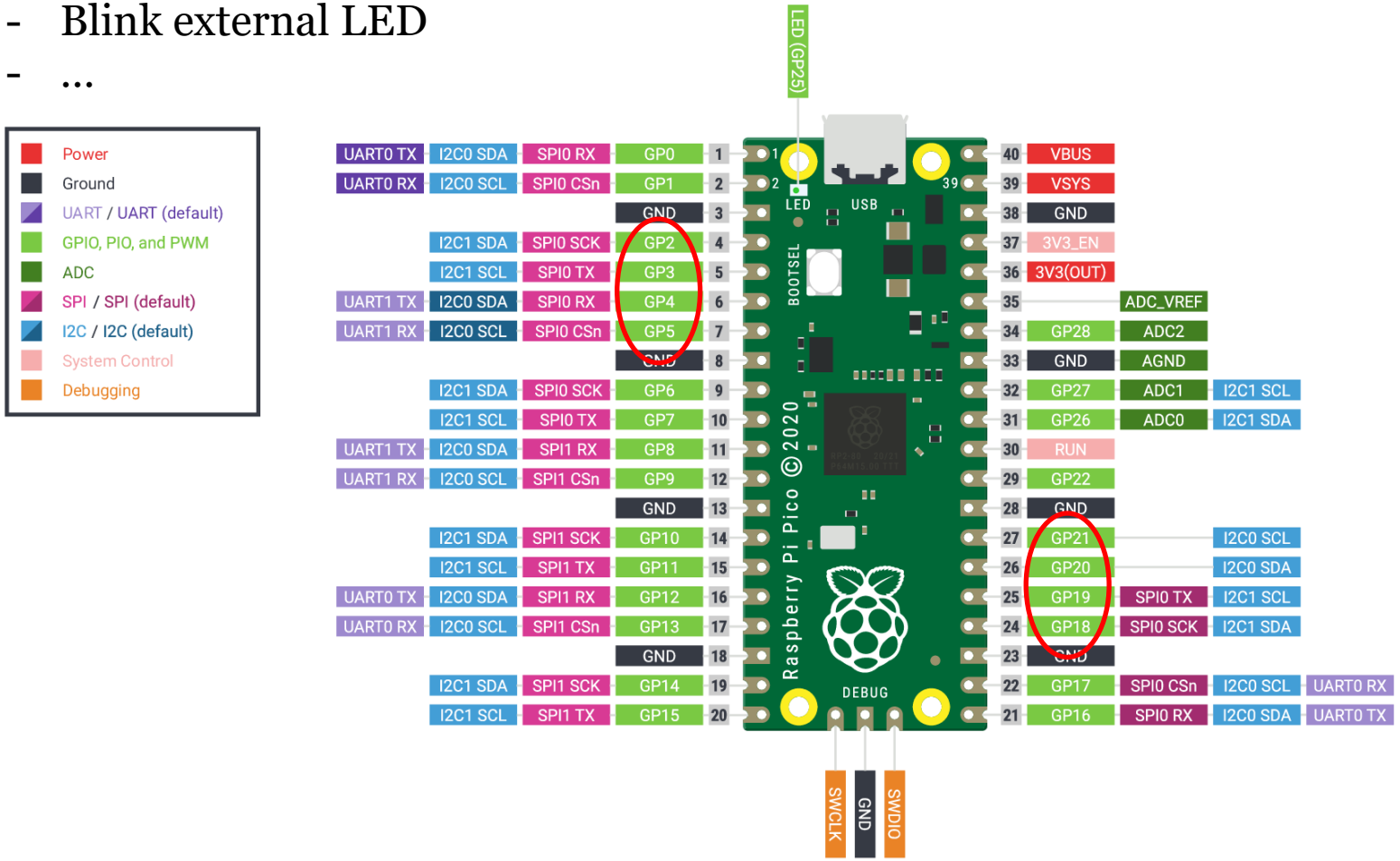


Raspberry Pi

Raspberry Pi Pico Pinout

GPIO (General Purpose Input Output)

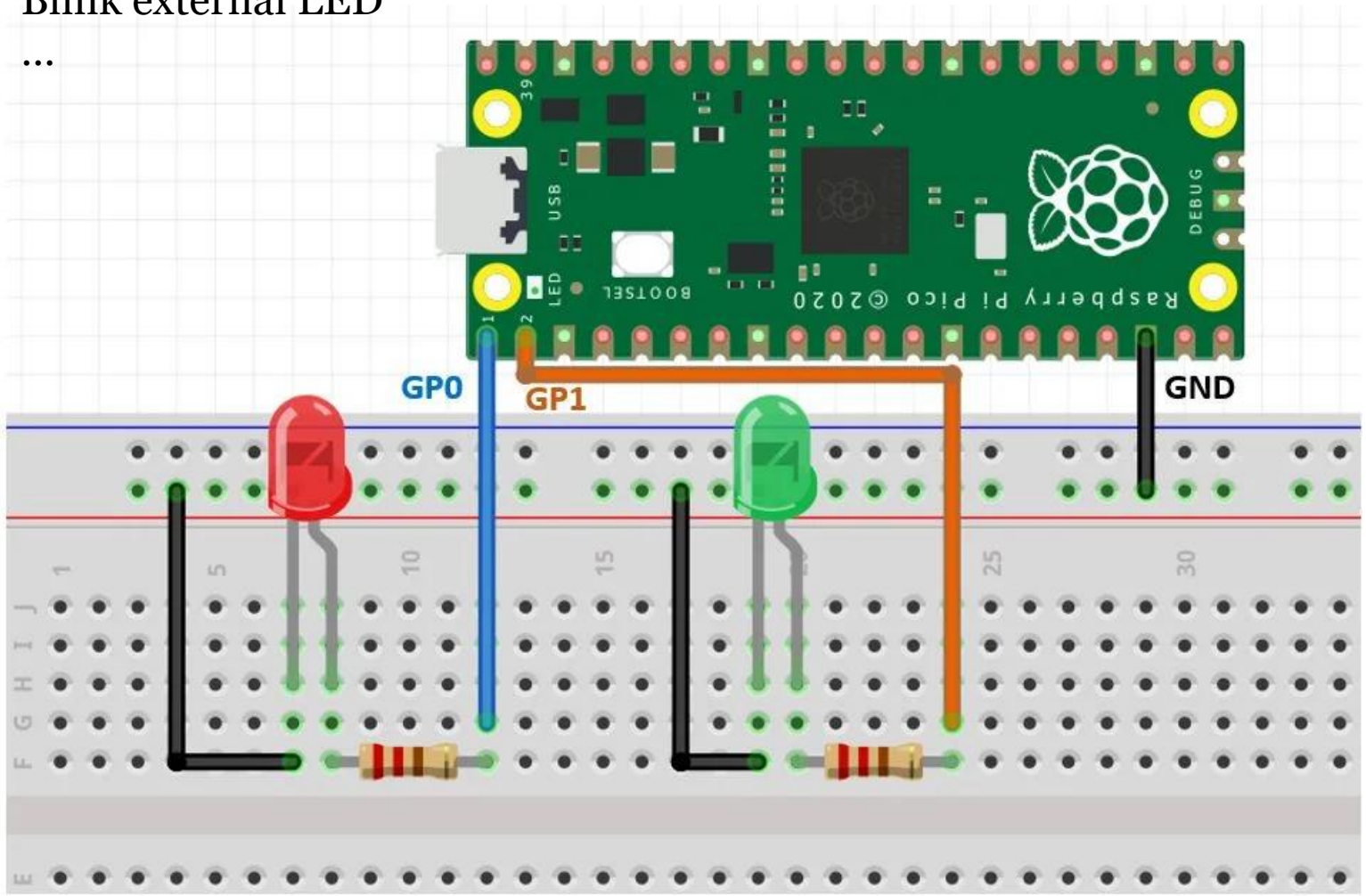
- Push buttons
- Blink external LED
- …



Pinout

GPIO (General Purpose Input Output)

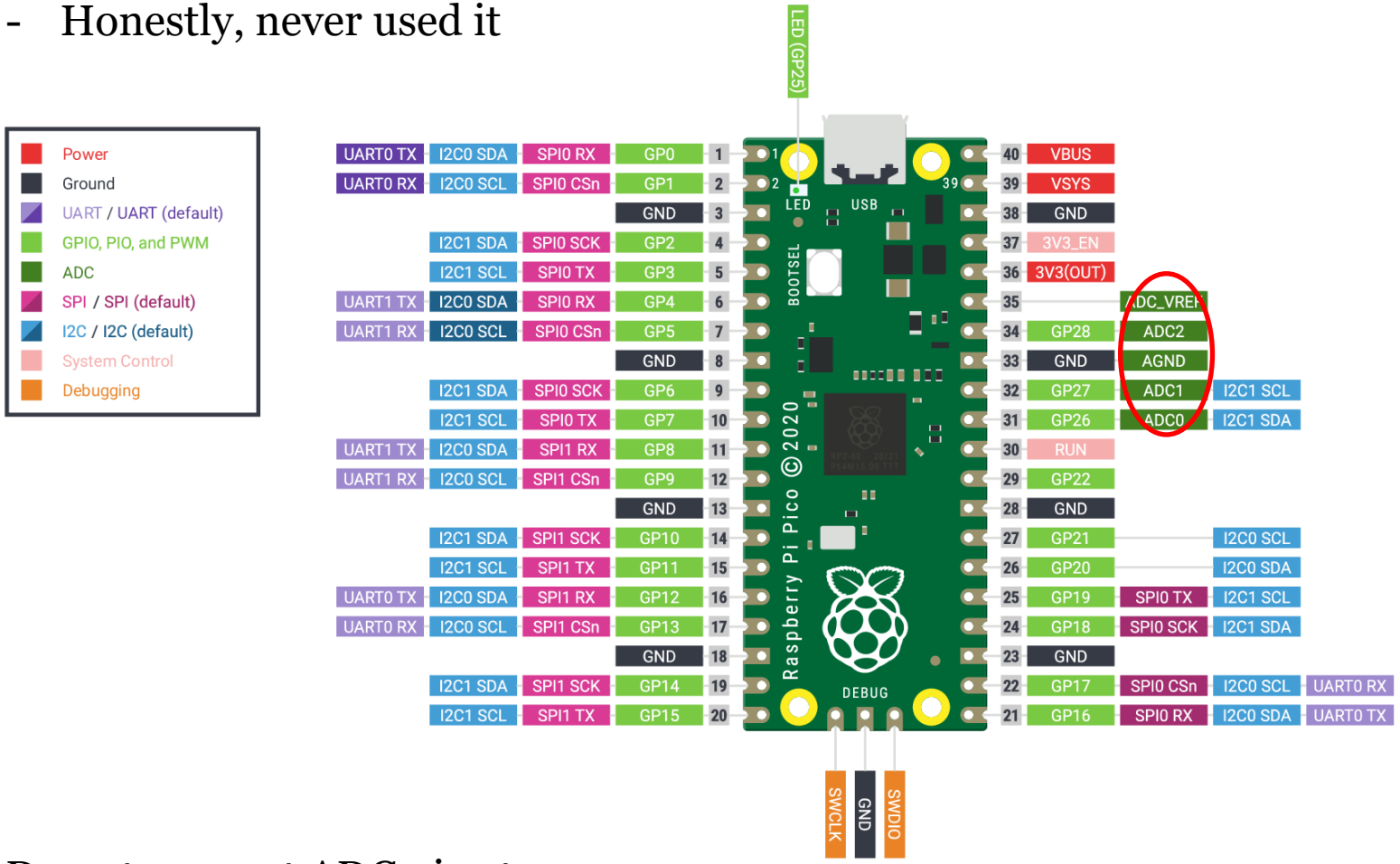
- Push buttons
- Blink external LED
- ...



Raspberry Pi Pico Pinout

ADC

- Read analog value from a sensor
- Honestly, never used it

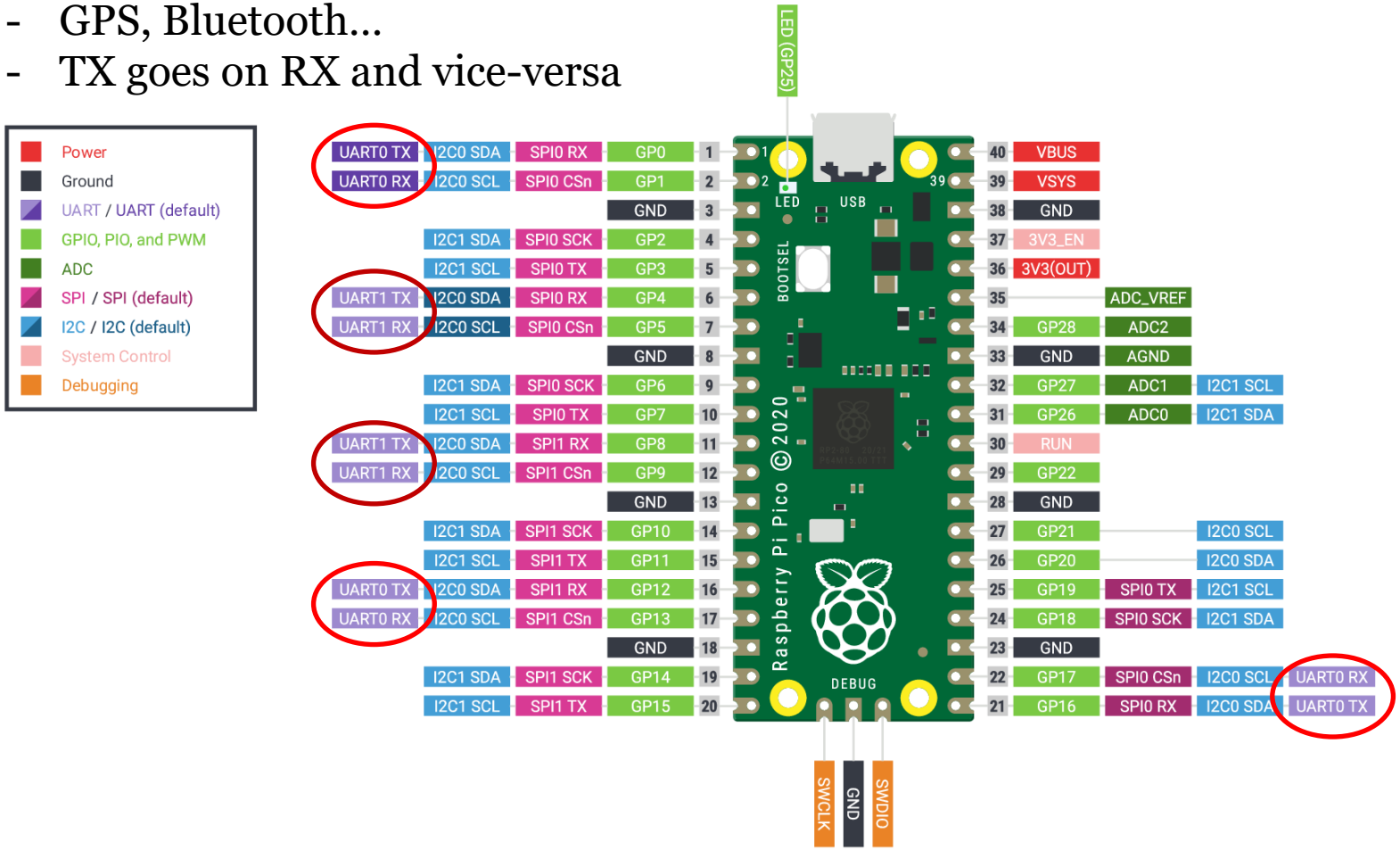


Do not connect ADC pins to more than 3.3V

Raspberry Pi Pico Pinout

UART (Universal Asynchronous Receiver/Transmitter)

- GPS, Bluetooth…
- TX goes on RX and vice-versa

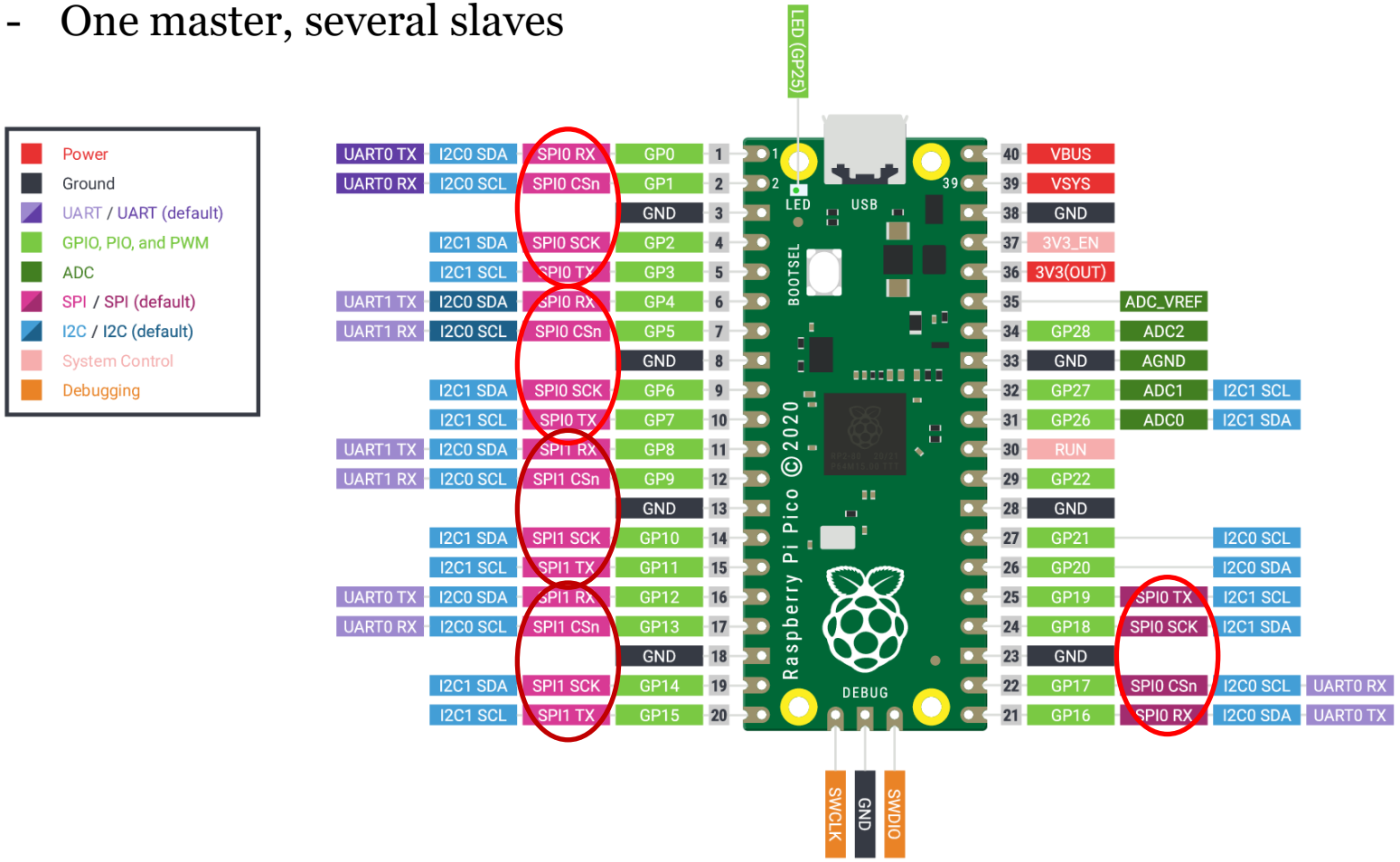


Raspberry Pi

Raspberry Pi Pico Pinout

SPI (Serial Peripheral Interface)

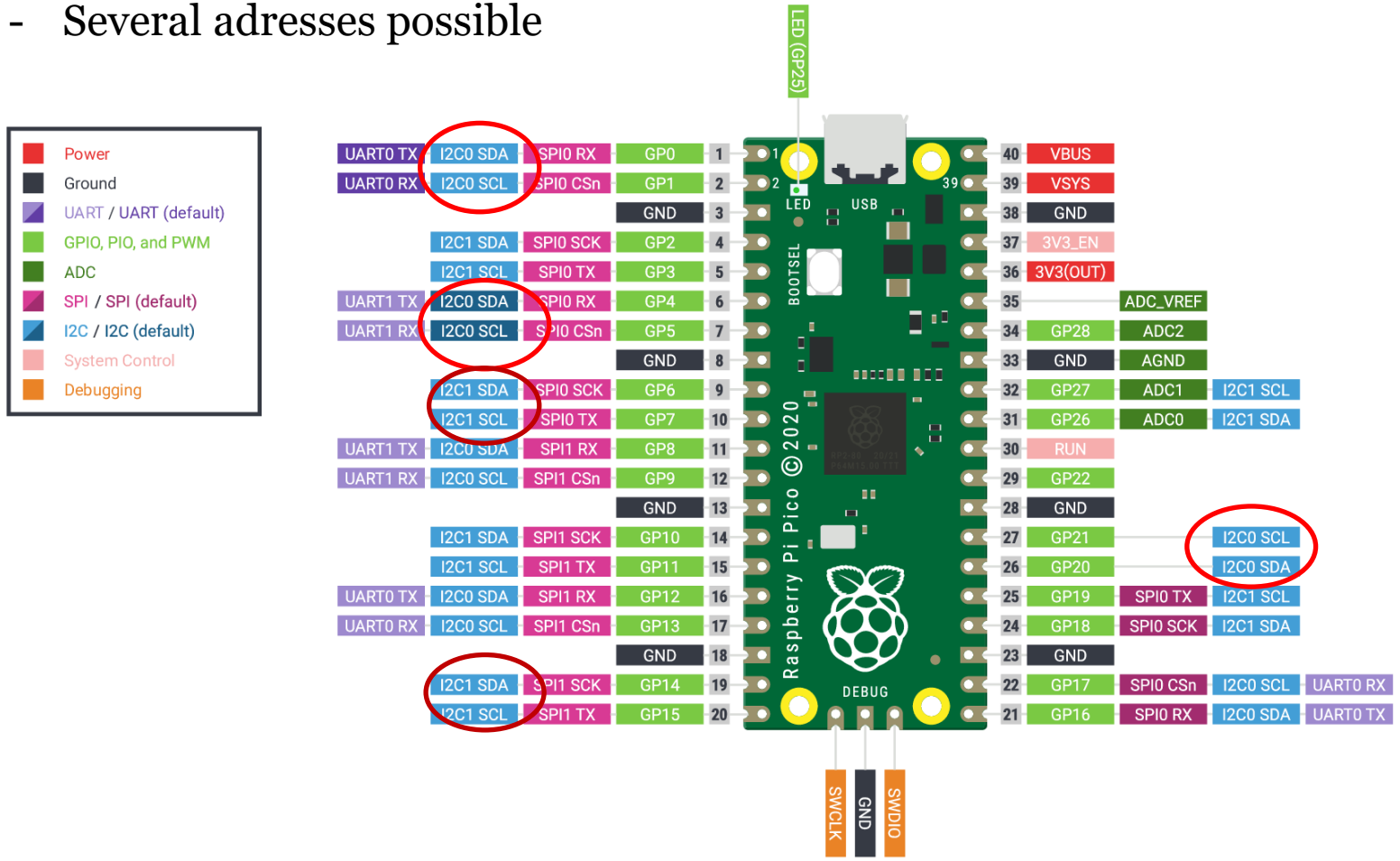
- LCD screens, sensors, memories…
- One master, several slaves



Raspberry Pi Pico Pinout

I2C (Inter-Integrated Circuit)

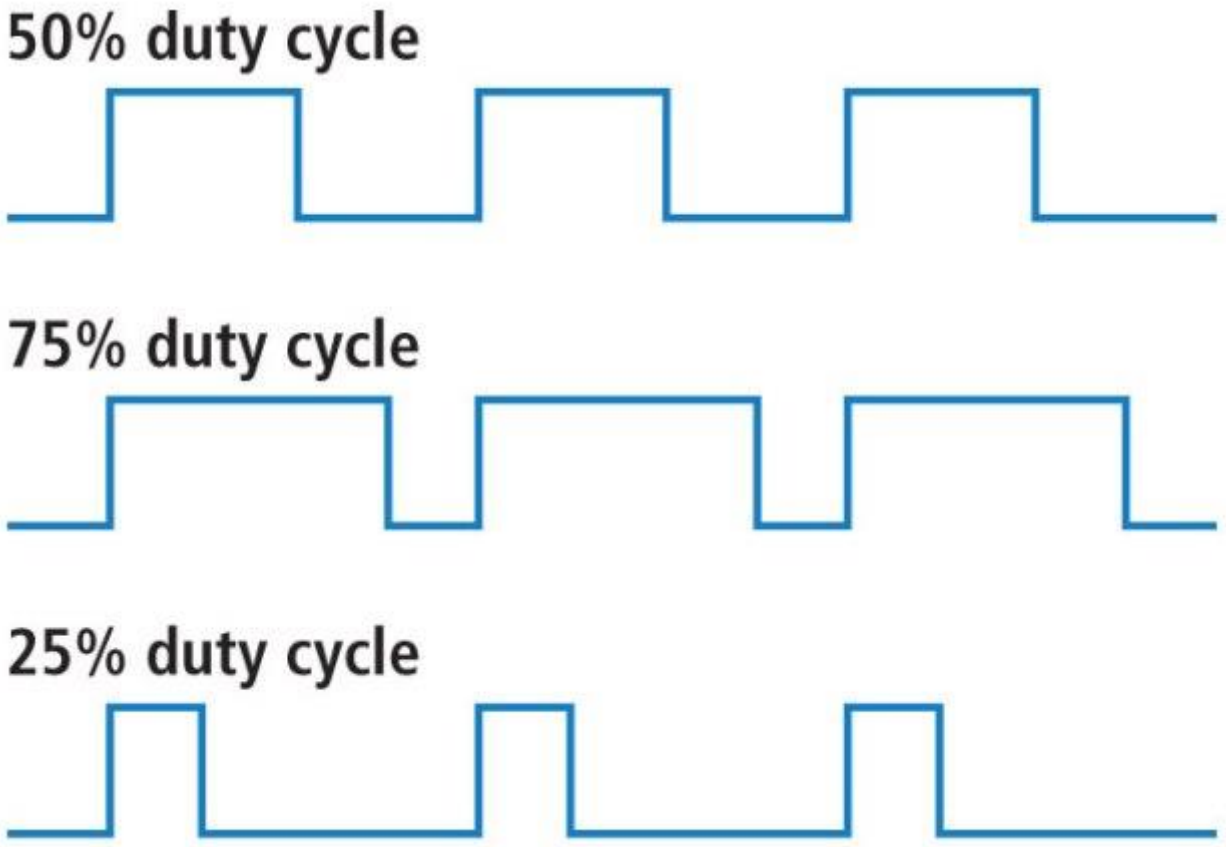
- LCD screens, sensors
- Several adresses possible



Pinout

PWM (Pulse Width Modulation)

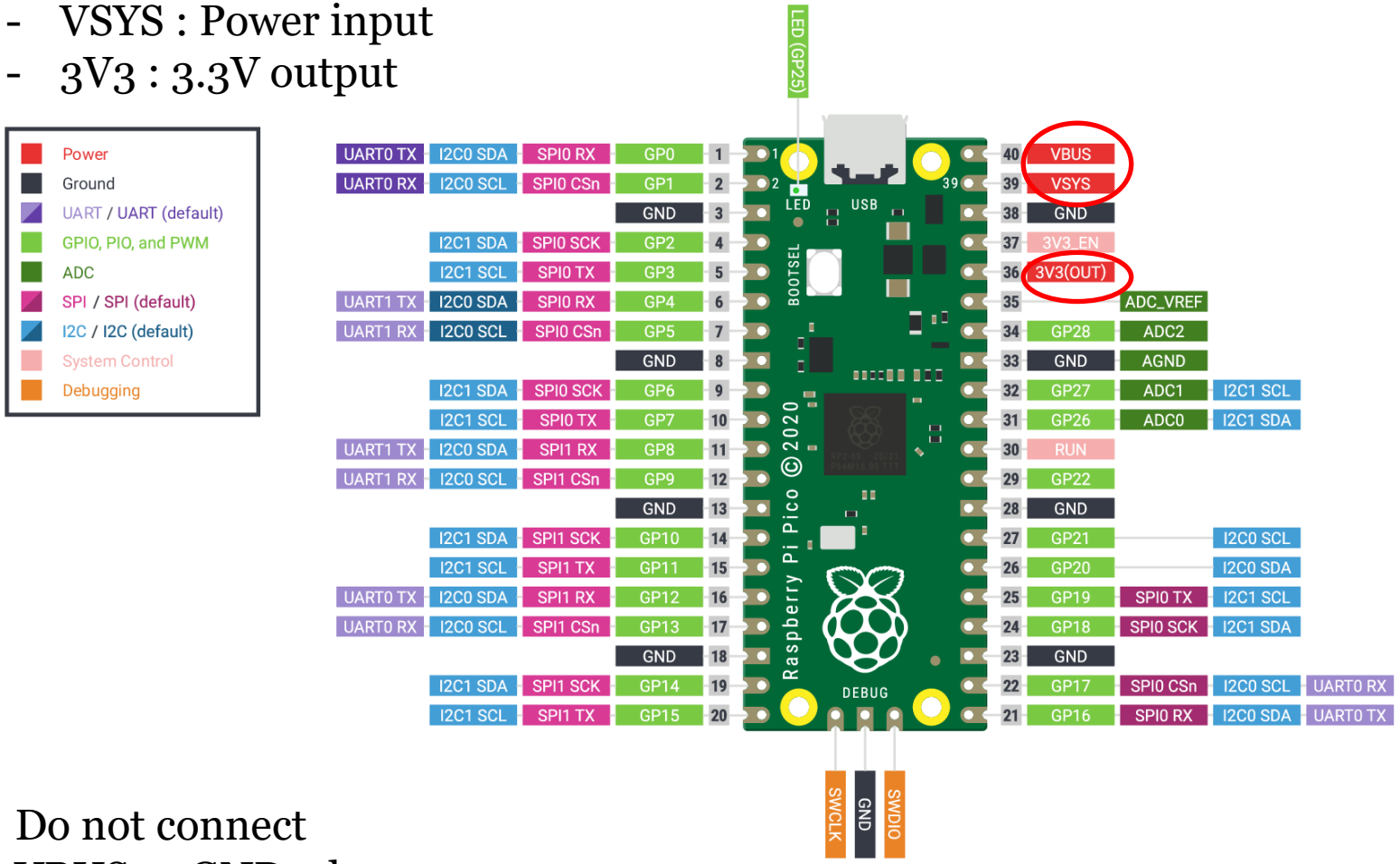
- Possible on all GPIO
- Used to construct a DC voltage with a square signal
- To control actuators, motors…



Raspberry Pi Pico Pinout

Power

- VBUS : 5V output when powered with USB
- VSYS : Power input
- 3V3 : 3.3V output



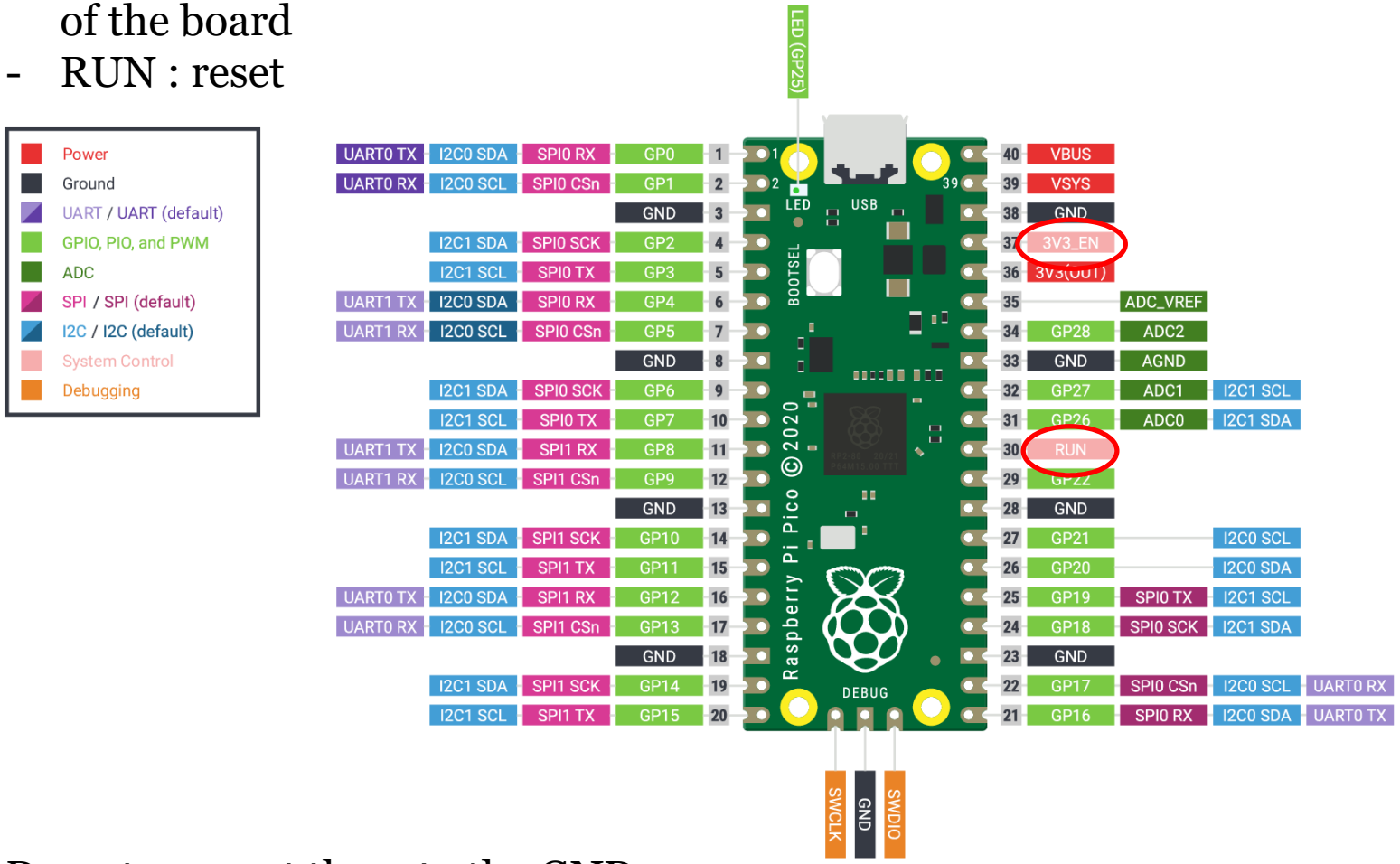
Do not connect VBUS to GND when powered with USB

Raspberry Pi

Raspberry Pi Pico Pinout

System Control

- 3V3_EN : controls the internal voltage regulator of the board
- RUN : reset

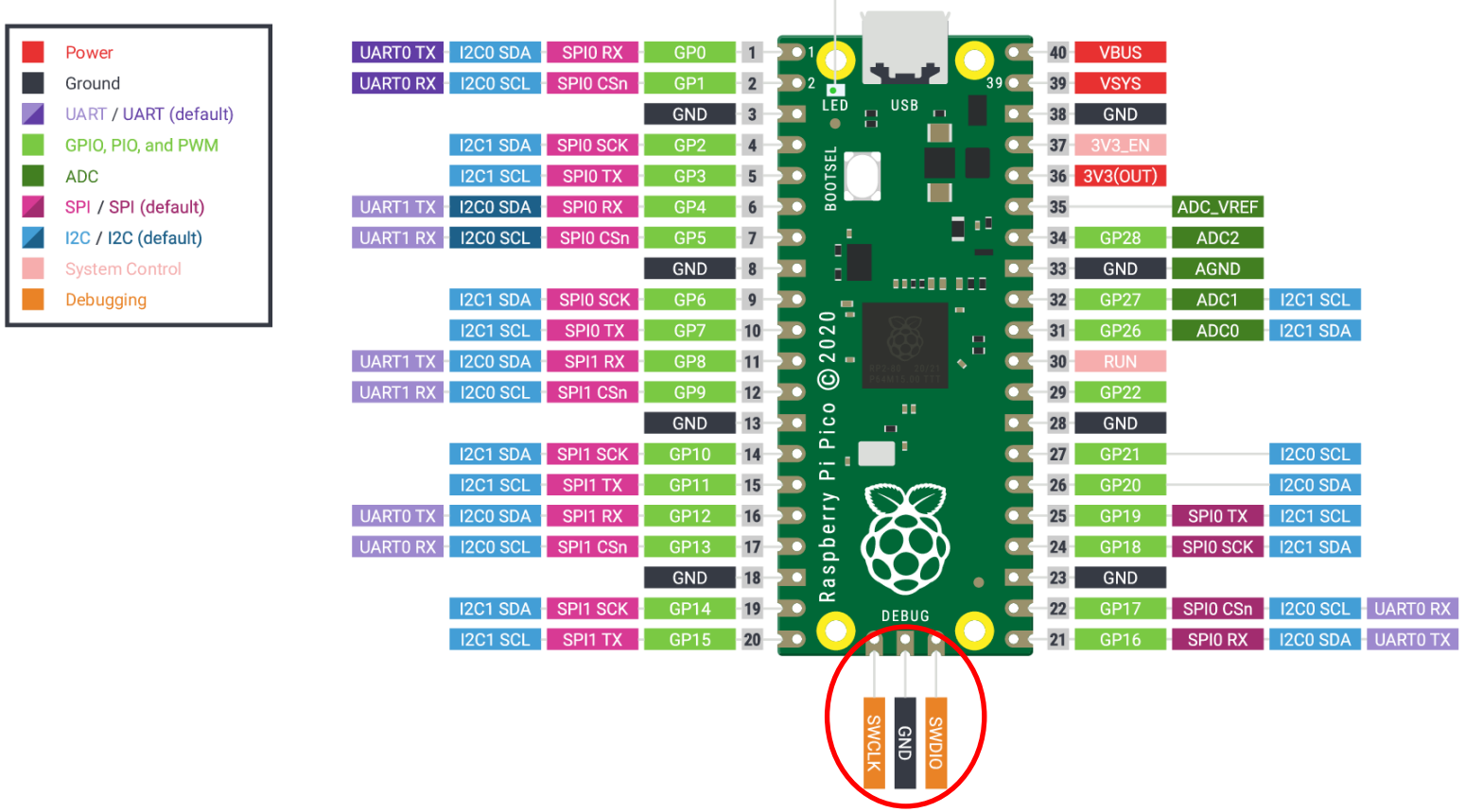


Do not connect them to the GND

Raspberry Pi Pico Pinout

SWD (Serial Wire Debug)

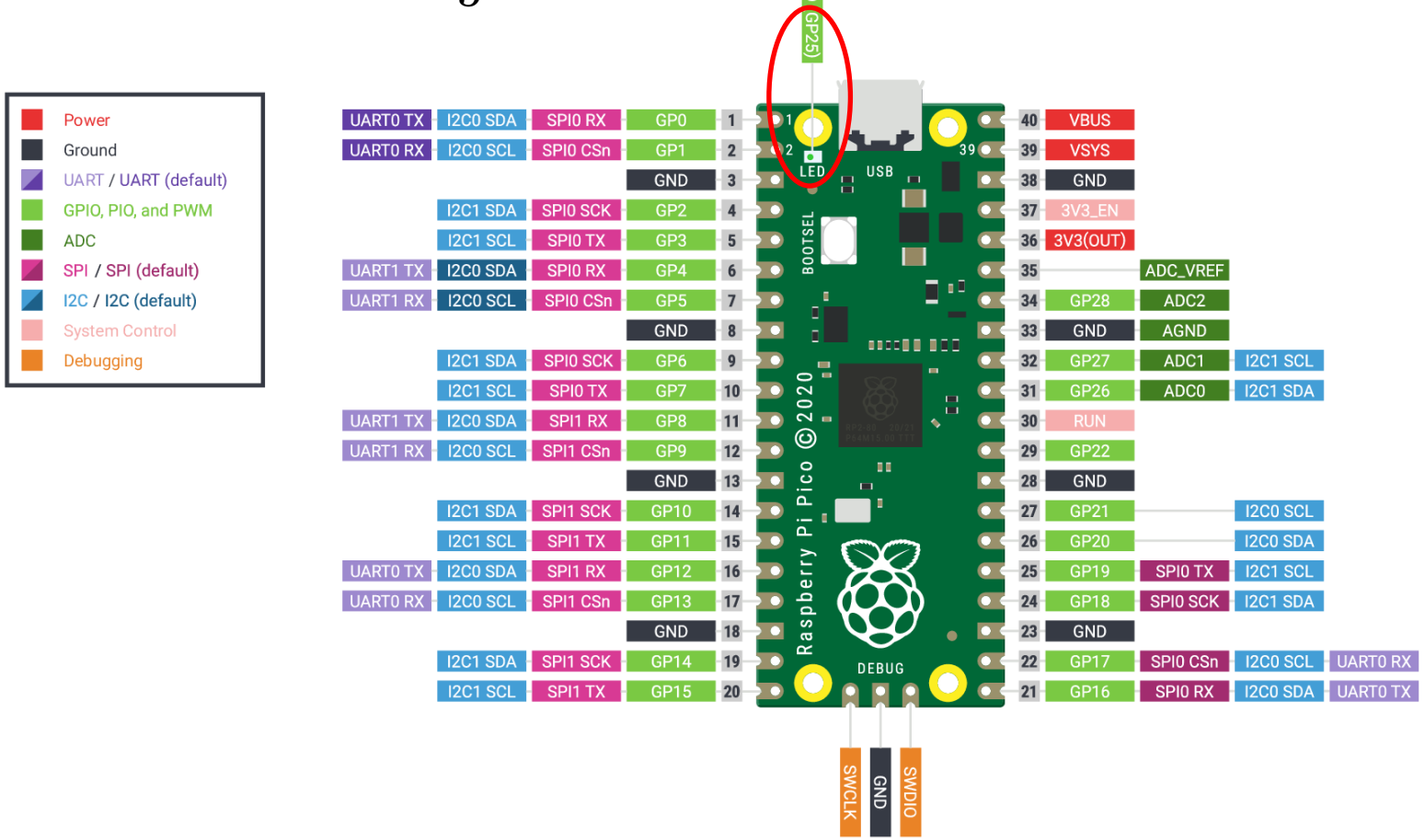
- SWDCLK and SWDIO
- You should not have to use them unless you are stupid enough to erase your flight data on your card (who said it happened to me last year ?)



Raspberry Pi Pico Pinout

What about the « missing » GPIO ?

- Internal GPIO
- For instance GPIO 25 for the internal LED



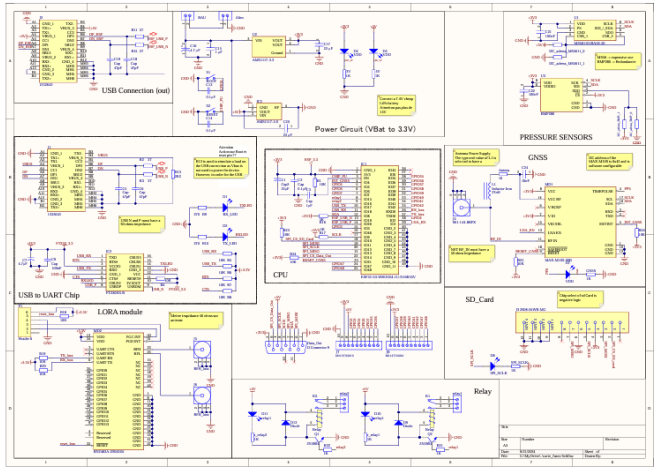
Raspberry Pi

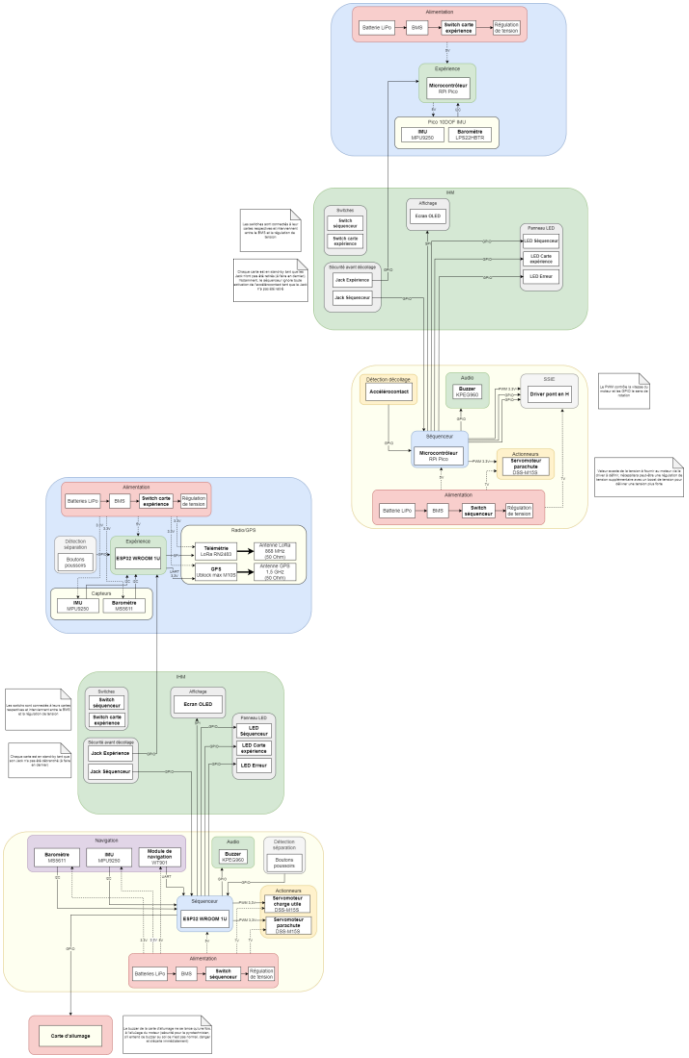
Presenting SCALAR 6

- Two-stage experimental rocket with two Cansats made by OSE on board
- Launch expected in July 2025 (C'Space)

- 6 boards with advanced functions (human-machine interface, navigation card, upper stage engine ignition…)

- Recruitment after Sparrow, if you're motivated by embedded systems you're welcome !





We'll see this during the lab on Saturday, you can have a look at the course document to download Thonny IDE.

Make sure all groups have a USB-A to micro USB connector that can transmit data.

Libraries

Python files containing functions, variables, classes...
Used not to have to reinvent everything for every project

To use a library, import it at the beginning of your program

```
import math  # Importe toute la librairie math
import os    # Importe toute la librairie os
```

Once imported, you can use the functions of the library with the syntax name_library.name_function().

```
import math

resultat = math.sqrt(16)  # Utilise la fonction sqrt de la
# librairie math pour calculer la racine carree de 16

print(resultat)  # Affiche 4.0
```

Also possible to import only a module or a function from a library

```
from math import sqrt  # Importe uniquement la fonction sqrt
                       # du module math

resultat = sqrt(25)
print(resultat)  # Affiche 5.0
```

Also possible to rename a library or a function when imported

```
import math as m  # Renomme math en m

resultat = m.sqrt(9)
print(resultat)  # Affiche 3.0
```

Some libraries are already installed on the Rpi Pico, to interact with devices and execute basic tasks

machine

Essential library to use the microcontroller, to control GPIOs, interfaces (I2C, SPI…), timers, PWM…

```
from machine import Pin

led = Pin(25, Pin.OUT)   # Cree un objet Pin pour controler la
                         # LED integree

led.value(1)   # Allume la LED
```

time

Manage delays... Used to pause a program or time events

```
import time

time.sleep(2)  # Pause de 2 secondes
print("2 secondes se sont ecoulees")
```

utime

Version of time specific to MicroPython, optimised for microcontrollers

```
import utime

start = utime.ticks_ms()  # Obtenir le temps actuel en
                          # millisecondes
utime.sleep_ms(100)       # Pause de 100 millisecondes
end = utime.ticks_ms()

print("Temps ecoule : ", utime.ticks_diff(end, start), "ms")
```

Classes

Create an object with specific variables (attributes) and functions (methods)

```
class Voiture:
    def __init__(self, marque, modele, annee):
        self.marque = marque
        self.modele = modele
        self.annee = annee

    def afficher_details(self):
        print(f"Voiture: {self.marque} {self.modele},
        Annee: {self.annee}")
```

```
# Creation d'une instance de la classe Voiture
ma_voiture = Voiture("Toyota", "Corolla", 2020)

# Appel de la methode afficher_details pour afficher les
# informations de la voiture
ma_voiture.afficher_details()
```

Some classes already installed on the RPi Pico with the machine library

Pin > Control GPIO

```
from machine import Pin

# Configuration d'une broche en sortie
led = Pin(25, Pin.OUT)
led.value(1)  # Allume la LED connectee a la broche 25

# Configuration d'une broche en entree
bouton = Pin(14, Pin.IN, Pin.PULL_DOWN)
etat_bouton = bouton.value()  # Lit l'etat du bouton (0 ou 1)
```

Some classes already installed on the RPi Pico with the machine library

ADC > Read analog values from sensors

```
from machine import ADC

# Creation d'un objet ADC sur la broche 26 (GP26)
potentiometre = ADC(26)

# Lecture de la valeur analogique (entre 0 et 65535)
valeur = potentiometre.read_u16()
print("Valeur lue :", valeur)
```

ADC.read_u16() returns a value between 0 and 65535 corresponding to the read voltage

Some classes already installed on the RPi Pico with the machine library

PWM > Generate PWM signals

```
from machine import Pin, PWM

# Configuration de la broche 15 en PWM
led_pwm = PWM(Pin(15))

# Reglage de la frequence du PWM
led_pwm.freq(1000)  # 1 kHz

# Reglage du rapport cyclique (duty cycle) entre 0 (eteint) et
# 65535 (100% allume)
led_pwm.duty_u16(32768)  # 50% de luminosite
```

Some classes already installed on the RPi Pico with the machine library

I2C > Interact with devices using I2C

```
from machine import Pin, I2C

# Configuration de l'I2C broches GPIO8 (SDA) et GPIO9 (SCL)
i2c = I2C(0, scl=Pin(9), sda=Pin(8), freq=400000)

# Scanner les peripheriques I2C connectes
devices = i2c.scan()
print("Peripheriques I2C trouves :", devices)

# Lire et ecrire des donnees sur un peripherique I2C
# Exemple : lire 2 octets a l'adresse 0x3C
data = i2c.readfrom(0x3C, 2)
```

Some classes already installed on the RPi Pico with the machine library

SPI > Interact with devices using SPI

```
from machine import Pin, SPI

# Configuration du SPI sur les broches GPIO10 (SCK), GPIO11
# (MOSI), GPIO12 (MISO)
spi = SPI(0, baudrate=1000000, polarity=0, phase=0,
          sck=Pin(10), mosi=Pin(11), miso=Pin(12))

# Ecriture et lecture de donnees SPI
# Exemple : ecrire [0x01, 0x02] et lire 2 octets
spi.write([0x01, 0x02])
data = spi.read(2)
```

Some classes already installed on the RPi Pico with the machine library

UART > Interact with devices using UART

```
from machine import UART

# Configuration du UART sur le port UART 0, avec les broches
# GPIO0 (TX) et GPIO1 (RX)
uart = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))

# Envoyer des donnees
uart.write('Hello, UART!')

# Lire des donnees
data = uart.read(10)  # Lit 10 octets de donnees
```

Writing data on your microcontroller


```
# Ouvrir (ou creer) un fichier nomme "donnees.txt" en mode
# ecriture
with open("donnees.txt", "w") as fichier:
    # Ecrire des donnees dans le fichier
    fichier.write("Bonjour, Raspberry Pi Pico !\n")
    fichier.write("Ceci est un exemple d'ecriture dans un
    fichier.\n")

# Le fichier est automatiquement ferme a la fin du bloc with
```

```
# Ouvrir le fichier en mode ajout
with open("donnees.txt", "a") as fichier:
    # Ajouter des donnees au fichier
    fichier.write("Ajout de nouvelles donnees.\n")
```

Be careful to open your file in append mode ‘a’ to add your data at the end of the file, and not in write mode ‘w’, which erases the data already written in the file. Use read mode ‘r’ to only read your data.

Writing data on your microcontroller

Use file.flush() each time data is written on the microcontroller to make sure it is immediately saved in the memory.

```
# Ouvrir un fichier en mode ecriture
fichier = open("donnees.txt", "w")

# Ecrire des donnees dans le fichier
fichier.write("Ecriture de donnees importantes.\n")

# Forcer l'enregistrement des donnees sur le disque
fichier.flush()

# Fermer le fichier
fichier.close()
```

Thank you for your attention ! Questions ?

Course documents in French and in English, as well as code sample can be found on the course GitHub repository.