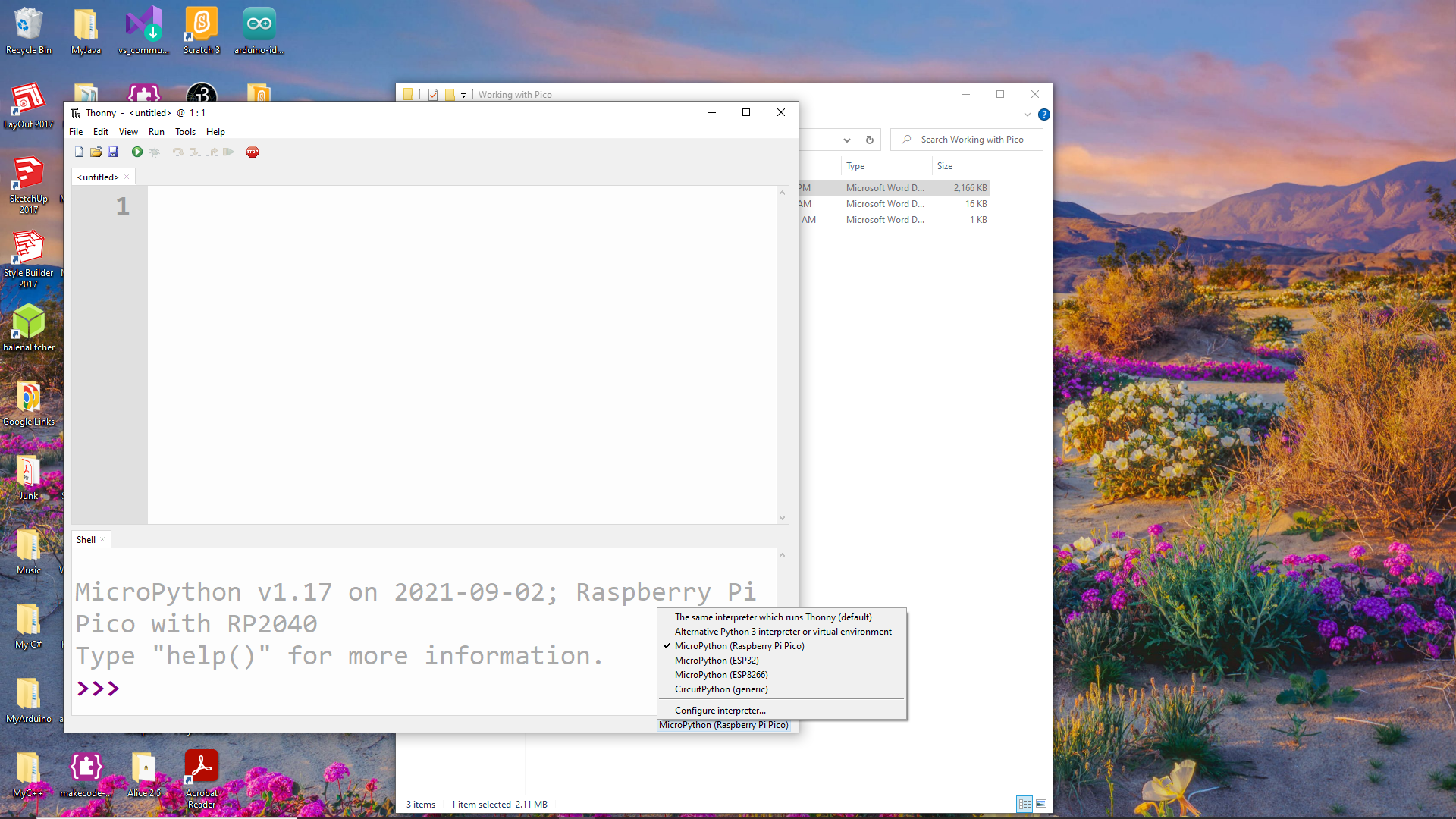
**Working With the Raspberry Pi Pico**

**Programming Using MicroPython**

For these activities, we will be using Thonny as our Integrated Development Environment (IDE). 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 separate Python installation, if necessary.) The initial user interface is stripped of all features that may distract beginners.

1. Plug in you Pico to a USB port.
2. Start Thonny.
3. In the lower right corner of the Thonny Window you can select the type of device you are going to be programming. In our case choose MicroPython (Raspberry Pi Pico).



1. Type in the following code:

# morse.py

# This program will convert a user provided phrase into Morse Code

# and flash an LED with the code

# Author: nmessa

# Date: 10.27.2021

from machine import Pin

import time

def codeGen(s):

output = ''

s = s.lower() #put string in lower case to reduce dictionary size

for i in range(len(s)):

if s[i] == '\n':

output += '\n'

continue

output += convert(s[i])

output += ' '

print (output)

for i in range(len(output)):

if output[i] == '.':

dot()

if output[i] == '-':

dash()

if output[i] == ' ':

time.sleep(0.1)

def convert(letter):

#International Morse Code dictionary

table = {'a':'.-', 'b':'-...', 'c':'-.-.', 'd':'-..', 'e':'.',

'f':'..-.', 'g':'--.', 'h':'....', 'i':'..', 'j':'.---',

'k':'-.-', 'l':'.-..', 'm':'--', 'n':'-.', 'o':'---',

'p':'.--.', 'q':'--.-', 'r':'.-.', 's':'...', 't':'-',

'u':'..-', 'v':'...-', 'w':'.--', 'x':'-..-', 'y':'-.--',

'z':'--..', ' ':' ', '1':'.----', '2':'..---',

'3':'...--', '4':'....-', '5':'.....', '6':'-....',

'7':'--...', '8':'---..', '9':'----.', '0':'-----'}

if letter in table.keys(): #check that letter is in dictionary

return table[letter]

else: #handle other characters where code does not exist in dictionary

return '' # 2 single quotes

def dot():

led.toggle()

time.sleep(0.1)

led.toggle()

time.sleep(0.1)

def dash():

led.toggle()

time.sleep(1)

led.toggle()

time.sleep(0.1)

#Main program

while True:

led = Pin(25, Pin.OUT)

phrase = input("Enter a phrase: ")

codeGen(phrase)

1. Run your program.
2. Create a new program in Thonny and type in the following code:

# Braille Translator

# Author: nmessa

# Date: 11.4.2022

# This program inputs user defined text and output Braille

# ASCII

asciicodes = [' ','!','"','#','$','%','&','','(',')','\*','+',',','-','.','/',

'0','1','2','3','4','5','6','7','8','9',':',';','<','=','>','?','@',

'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q',

'r','s','t','u','v','w','x','y','z','[','\\',']','^','\_']

# Braille symbols

brailles = ['⠀','⠮','⠐','⠼','⠫','⠩','⠯','⠄','⠷','⠾','⠡','⠬','⠠','⠤','⠨','⠌',

'⠴','⠂','⠆','⠒','⠲','⠢','⠖','⠶','⠦','⠔','⠱','⠰','⠣','⠿','⠜','⠹',

'⠈','⠁','⠃','⠉','⠙','⠑','⠋','⠛','⠓','⠊','⠚','⠅','⠇','⠍','⠝','⠕',

'⠏','⠟','⠗','⠎','⠞','⠥','⠧','⠺','⠭','⠽','⠵','⠪','⠳','⠻','⠘','⠸']

#Create Braille conversion dictionary

convert = {}

#Build dictionary

for i in range(len(asciicodes)):

convert[asciicodes[i]] = brailles[i]

phrase = input("Enter a phrase: ").lower() #convert to only lower case

#create empty Braille string

braille = ""

#convert each letter in the phrase into Braille

for letter in phrase:

braille += convert[letter]

#print the new Braille string

print(braille)

1. Run your program.

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated

It’s time to make the LED blink. To do that, you’ll need to create a loop. Write the following

program so it matches the one below:

**import machine**

**import utime**

**led\_onboard = machine.Pin(25, machine.Pin.OUT)**

**while True:**

**led\_onboard.value(1)**

**utime.sleep(5)**

**led\_onboard.value(0)**

**utime.sleep(5)**

Alternatively

**import machine**

**import utime**

**led\_onboard = machine.Pin(25, machine.Pin.OUT)**

**while True:**

**led\_onboard.toggle()**

**utime.sleep(5)**

Using a breadboard

If you’re using a breadboard to hold the components and make the electrical connections.

A breadboard is covered with holes – spaced, to match components, 2.54 mm apart. Under

these holes are metal strips which act like the jumper wires you’ve been using until now. These

run in rows across the board, with most boards having a gap down the middle to split them in

two halves.

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Using an external LED**

Graphical user interface, text, application

Description automatically generated

**import machine**

**import utime**

**led\_external = machine.Pin(15, machine.Pin.OUT)**

**while True:**

**led\_external.toggle()**

**utime.sleep(5)**

**Inputs: reading a button**

Graphical user interface, text

Description automatically generated

import machine

import utime

button = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL\_DOWN)

while True:

if button.value() == 1:

print("You pressed the button!")

utime.sleep(2)

**Inputs and outputs: putting it all together**

Graphical user interface, text, application, Word

Description automatically generated

import machine

import utime

led\_external = machine.Pin(15, machine.Pin.OUT)

button = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL\_DOWN)

while True:

if button.value() == 1:

led\_external.value(1)

utime.sleep(2)

led\_external.value(0)

**Traffic Light Controller**

Graphical user interface, application, Word

Description automatically generated

import machine

import utime

led\_red = machine.Pin(15, machine.Pin.OUT)

led\_amber = machine.Pin(14, machine.Pin.OUT)

led\_green = machine.Pin(13, machine.Pin.OUT)

while True:

led\_green.value(1)

utime.sleep(5)

led\_green.value(0)

led\_amber.value(1)

ytime.sleep(2)

led\_amber.value(0)

led\_red.value(1)

utime.sleep(5)

led\_red.value(0)

**Advanced Traffic Light System**

Graphical user interface, application, Word

Description automatically generated

import machine

import utime

import \_thread

led\_red = machine.Pin(15, machine.Pin.OUT)

led\_amber = machine.Pin(14, machine.Pin.OUT)

led\_green = machine.Pin(13, machine.Pin.OUT)

button = machine.Pin(16, machine.Pin.IN, machine.Pin.PULL\_DOWN)

buzzer = machine.Pin(12, machine.Pin.OUT)

global button\_pressed

button\_pressed = False

def button\_reader\_thread():

global button\_pressed

while True:

if button.value() == 1:

button\_pressed = True

utime.sleep(0.01)

\_thread.start\_new\_thread(button\_reader\_thread, ())

while True:

led\_green.value(1)

utime.sleep(5)

led\_green.value(0)

led\_amber.value(1)

utime.sleep(2)

led\_amber.value(0)

if button\_pressed == True:

led\_red.value(1)

for i in range(10):

buzzer.value(1)

utime.sleep(0.2)

buzzer.value(0)

utime.sleep(0.2)

led\_red.value(0)

global button\_pressed

button\_pressed = False

else:

led\_red.value(1)

utime.sleep(5)

led\_red.value(0)