RP2040 hands-on

Form a group and complete these tasks

* Create a folder on google drive for your batch
* Upload the screenshots and program in that folder
* Always save a copy of the simulations in wokwi and then save with your preferred name.
* Ctrl + Z will not work in wokwi, don’t use it.
* Learn the programming, it is very important.
* Try other simulations also, don’t worry about library files, you can create them yourself. (copy paste from already available examples and name the library files with same name)

Task 1

# Getting the 7 segment LED to work!

Link: <https://wokwi.com/projects/300210834979684872>

Always save a copy before editing the simulation

## Intended Outcome

* You will understand how to program a 7 segment LED
* You will learn how to use python program to control the 7 segment display

## Points to remember

1. Learn about the 7 Segment Display
   1. They are the normal LEDs but in a specific pattern made to like the figure 1.
   2. Like how you connect power supply to LED and make them glow, you make 7 segment LED to glow as well, however we have to see the type of the LED. In out case (WOKWI) we have used a common Anode mode LED. This implied that the LED input must be low to make it glow.

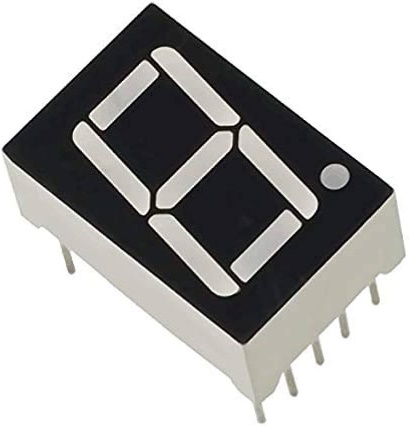


Figure 1 7 Segment LED

* 1. Hence to glow LED we give input as “0” in the program ( 0 is digital and its analog equivalent is 0V and 1 is the digital and its analog equivalent is 3.3V) hence in program we use 0 and 1 to trigger the LED’s on and off.
  2. And this LED’s in 7 segment LED are arranged in this way as shown in figure 2.
     1. As from diagram we can understand there are 7 LED’s to form a character and a decimal pointer.
     2. If the LED A, B,C,D,G are made to glow, then you can see character 3, right?

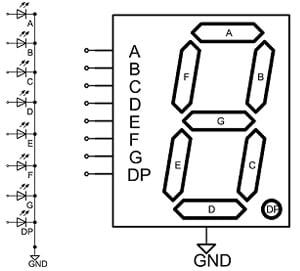


Figure 2 a. LED's in 7 Segment LED b.7 segment LED displaying 3.



1. Getting started with the programming
   1. Go to this Link: <https://wokwi.com/projects/300210834979684872>
   2. See the programming and observe the connections, simulate and see the results
   3. The program is written in python, to display the images in ascending order
   4. There is switch connected, when its toggled the number display is in descending order.

The program is here :

from machine import Pin

Initialization of library

Initialization of library

from utime import sleep

# 7-segment display layout

#       A

How the LED’s are arranged (no DP here)

#      ---

#  F |  G  | B

#      ---

#  E |     | C

#      ---

#       D

pins = [

    Pin(2, Pin.OUT),  # A

Telling the RP 2040 that these pins are the output

    Pin(3, Pin.OUT),  # B

    Pin(4, Pin.OUT),  # C

    Pin(5, Pin.OUT),  # D

    Pin(6, Pin.OUT),  # E

    Pin(8, Pin.OUT),  # F

    Pin(7, Pin.OUT),  # G

    Pin(0, Pin.OUT)   # DP (not connected)

]

# Common anode 7-segment display digit patterns

digits = [

    [0, 0, 0, 0, 0, 0, 1, 1], # 0

    [1, 0, 0, 1, 1, 1, 1, 1], # 1

    [0, 0, 1, 0, 0, 1, 0, 1], # 2

    [0, 0, 0, 0, 1, 1, 0, 1], # 3

Assigning digit values, i.e what all LED’s must be turned on to display the characters starting from 0 all the way upto f (hexadecimal)

    [1, 0, 0, 1, 1, 0, 0, 1], # 4

    [0, 1, 0, 0, 1, 0, 0, 1], # 5

    [0, 1, 0, 0, 0, 0, 0, 1], # 6

    [0, 0, 0, 1, 1, 1, 1, 1], # 7

    [0, 0, 0, 0, 0, 0, 0, 1], # 8

    [0, 0, 0, 1, 1, 0, 0, 1], # 9

    [0, 0, 0, 1, 0, 0, 0, 1], # a

    [1, 1, 0, 0, 0, 0, 0, 1], # b

    [0, 1, 1, 0, 0, 0, 1, 1], # C

    [1, 0, 0, 0, 0, 1, 0, 1], # d

    [0, 1, 1, 0, 0, 0, 0, 1], # E

    [0, 1, 1, 1, 0, 0, 0, 1], # F

]

def reset():

    """Turns off all segments on the 7-segment display."""

    for pin in pins:

        pin.value(1)

reset()

Assigning the **switch** to pin 13 and telling RP 2040 that it is an input

Assigning the switch to pin 13 and telling RP 2040 that it is an input

switch = Pin(13, Pin.IN)

while True:

    if switch.value() == 1:

        # Ascending counter

First starting with the **switch** position, if it on

Then for all the length of the given input start decreasing the value but stop and change if the **switch** position is changed. If the switch position is changed the numbers will be displayed in opposite order

        for i in range(len(digits)):

            if switch.value() == 0:

                break

            for j in range(len(pins) - 1):

                pins[j].value(digits[i][j])

            sleep(0.5)

    else:

        # Descending counter

        for i in range(len(digits) - 1, -1, -1):

            if switch.value() == 1:

                break

            for j in range(len(pins)):

                pins[j].value(digits[i][j])

            sleep(0.5)

## Now to do

1. Redesign the diagram (if necessary) and display your name in single 7 segment LED (I tried mine and result in <https://wokwi.com/projects/371049434220144641>) certain alphabets are not possible to display so don’t worry ignore ‘em.
2. Take a screenshot of the results, create a drive folder and save it! We will assess it later.

Task 2

# Advancing the display to 16x2 LCD !

Link: <https://wokwi.com/projects/359400194112248833>

## Intended Outcome

* You will get to know what is 16x2 LCD display.
* You will learn how to use python program to display anything using 16x2 LCD display

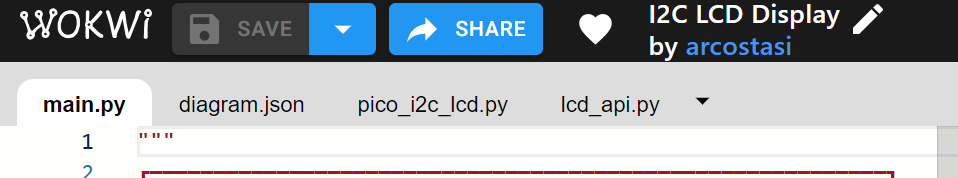
## Points to remember

1. Let’s start with the LCD display
2. Hope you all know what registers are? We have discussed them a lot in class. Here we are going to know about these registers
3. Like RP2040 having registers, the LCD also have its own set of registers see in figure 3
   1. Accessing these registers will tell the LCD what to do whether,
      1. To treat the information as command (eg clear screen, blink cursor) or
      2. To treat them as data (eg. Displaying the words)



Figure 3 A 16x2 LCD

1. Open the link <https://wokwi.com/projects/359400194112248833> and go to the lcd\_api.py tab



1. In this tab you can see how the registers are assigned with code words to help us program them easily. To learn more about these go to this link: <https://www.electronicsforu.com/technology-trends/learn-electronics/16x2-lcd-pinout-diagram#:~:text=A%2016X2%20LCD%20has%20two,instructions%20given%20to%20the%20LCD.>
2. Now check the connections and observe the code and try to modify then and there and see how the outputs vary.
3. Here, you can see that we are using an I2C protocol to enable communication between the raspberry pico and the LCD display. That is we have used only 4 pins (2 meant for power supply) and another 2 for I2C, the SDA and SCL (Serial DAta and Serial CLock)
4. Let us see the program now

Initializing the library / see the wokwi docs for more information

from machine import I2C, Pin

from time import sleep

from pico\_i2c\_lcd import I2cLcd

def initialize\_i2c\_lcd(sda\_pin, scl\_pin, i2c\_freq):

Writing a function for initializing LCD, since its through I2C mode, SCL and SDA pins are included

Writing a function for initializing LCD, since its through I2C mode, SCL and SDA pins are included

    """Initialize the I2C LCD display with the given parameters."""

    i2c\_bus = I2C(0, sda=Pin(sda\_pin), scl=Pin(scl\_pin), freq=i2c\_freq)

    i2c\_address = i2c\_bus.scan()[0]

    return I2cLcd(i2c\_bus, i2c\_address, 2, 16), i2c\_address

def display\_address(lcd, i2c\_address):

    """Display the I2C address in decimal and hexadecimal formats."""

    lcd.blink\_cursor\_on()

    for address\_format in (str, hex):

Function to find the address of the display (remember like telling your friend where fridge is in your house.)

Function to find the address of the display (remember like telling your friend where fridge is in your house.)

        lcd.putstr(f"I2C Address: {address\_format(i2c\_address)}\n")

        lcd.putstr("PI PICO Hardware")

        sleep(2)

        lcd.clear()

    lcd.blink\_cursor\_off()

def backlight\_test(lcd):

    """Perform a backlight test by blinking the backlight 10 times."""

    lcd.clear()

Function to perform an operation that is backlight test, Here the backlight is made to blink 10 times, as we know it is very boring to write same set of instructions 10 times to blink 10 times, we write a function. (function is like telling your friend where fridge is, once you tell him you don’t have to tell him again where the fridge is. Next time when you want something from fridge you tell him to get it without telling where fridge is!

    lcd.putstr("Backlight Test")

    for \_ in range(10):

        lcd.backlight\_on()

        sleep(0.2)

        lcd.backlight\_off()

        sleep(0.2)

    lcd.backlight\_on()

def hide\_cursor\_count(lcd):

    """Hide the cursor and display a count from 0 to 19."""

    lcd.clear()

    lcd.hide\_cursor()

    for count in range(20):

        lcd.putstr(str(count))

        sleep(0.4)

        lcd.clear()

def main():

    """Main function to run the I2C LCD display example."""

    lcd\_display, i2c\_address = initialize\_i2c\_lcd(sda\_pin=0, scl\_pin=1, i2c\_freq=400000)

    while True:

        display\_address(lcd\_display, i2c\_address)

        backlight\_test(lcd\_display)

        hide\_cursor\_count(lcd\_display)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

## Now to do

1. Redesign the diagram (if necessary) and display your name/favourite songs or wtv you wish to display (I tried mine and result in <https://wokwi.com/projects/371052274327113729>)
2. Take a screenshot of the results, create a drive folder and save it! We will assess it later.
3. Try OLED display (<https://wokwi.com/projects/359558101922696193>)
4. Write program to add/sub/divide/multiply and display the output in OLED
5. Write program to perform logical operations(AND,OR,NAND,NOT) and display in LCD.

Task 3

# Dealing with analog world

Link: https://wokwi.com/projects/371048832052288513

## Intended Outcome

* You will get to why we need ADC
* You will learn how to use python program to convert those analog value to manipulate digitally.

## Points to remember

1. ADC is the analog to digital convertor. It is necessary because most of the sensors we use senses analog quantity like light, temperature, pressure and so on. If you need to build an automatic system then these sensors must be hooked with a microprocessor, thus making ADC a must for them.
2. There are different types of ADC ( we have studied the basic ones in LIC, …. I hope)
3. We use SAR type ADC (see page number 562 in the PR2040 datasheet)
4. Now go to the link <https://wokwi.com/projects/371048832052288513> and perform the simple simulation (you can adjust the potentiometer in the simulator)
5. Ok! Here how we are saying this as ADC is,
   1. The voltage is Analog right?
   2. We have taken the voltage and given it to a variable resistor, that is the potentiometer
   3. When the potentiometer knob is changed, the resistance changes and the voltage output change
   4. We are going to convert these voltage changes into digital values
   5. How we do it is, the RP2040 has in built ADC which can be accessed in any pins between 26 to 29 (4 dedicated pint)
   6. In the simulator pin 26 is used
6. Now observe the simulation and try to mimic it in new file.
7. Hope I don’t have to explain the program

## Now to do

1. Open a new file, create the same setup and then
   1. Add a LED that should glow if the potentiometer knob position is varied.

Sample: <https://wokwi.com/projects/371054811470068737>

* 1. Add a motor and control it using the potentiometer
  2. Go back to the previous task and add these both potentiometer and motor, display the motors status in the OLED, the output should be like “Motor is On” or “Motor is Off”
  3. Save the file offline for further assessment.