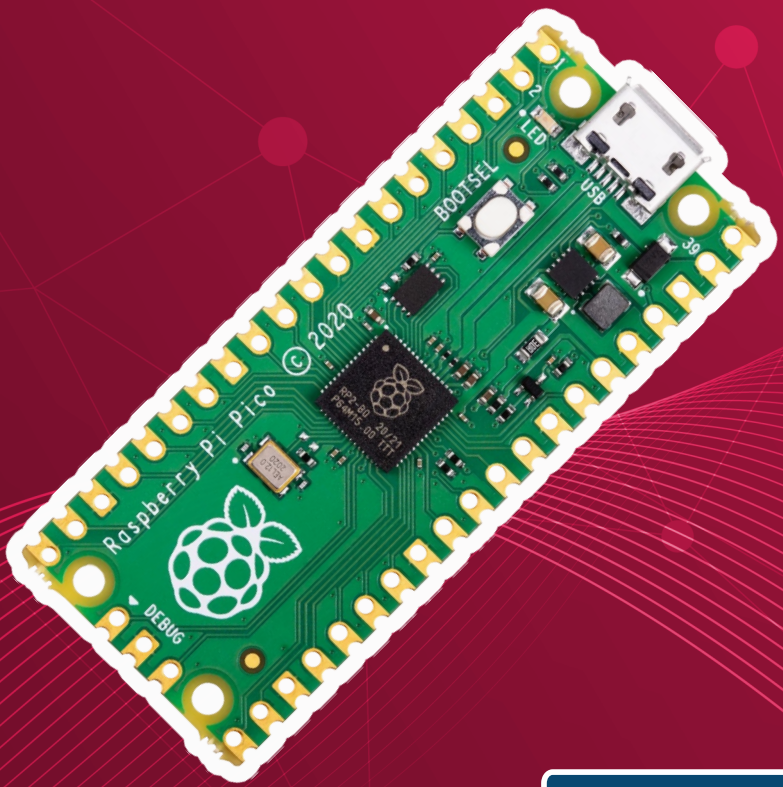
Raspberry Pi Pico Project Book



robotistan

Welcome to the World of Electronics and Coding. Now that you have opened this book,you are eager to swin in the sea of wonder and learn new things.

Although learning new things in such matters is thought to be difficult, if you proceed step by step and with the right practices, you will realize that it is very simple. As long as the applications are made in the early stages, there will be places that do not sit down.You will overcome this problem as you practice.

It just takes a little patience, so you can learn robotic programming with an easy and correct roadmap, starting from the easy and moving towards more complex. If you want to watch more detailed video explanations of the applications, you can go to our Youtube channel by scanning the QR code at the back of the book. You can Access the codes writen in the booklet both from the description section of the related videos and from our blog page.

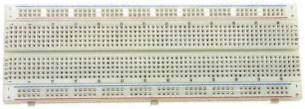
You can send us your set contents, applications, videos and any suggestions and questions you have in mind at "**info@robotistan.com**"

.

Robotistan Team

Contents

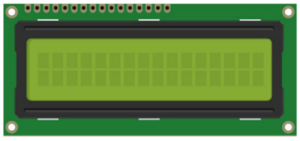
Raspberry Pi Pico:
Microcontroller we will perform our projects



Breadboard
It is our tool to test our circuits on the breadboard. It allows us to easily test the circuits we have built without soldering them to each other. It allows us to test the circuits we have designed be-fore transferring them to printed circuit boards or perforated pla-tes.

HC-SR501 PIR Sensor
PIR sensors are sensors used to detect the movement of living things in an environment. This tiny sensor is a module that can be used with many mic-rocontroller platforms such as Raspberry Pi, Arduino, ESP32, which you can use comfortably in various electronics, robotics and hobby applications.
This module, which has a digital output, gives a logic 0 output when it does not detect motion in the environment, and a logic 1 output when it detects motion.

Resistor:
In electrical circuits, resistance is the strain faced by an electric current flowing through a conductor.



2X16 LCD Screen:
It is a screen consisting of 2 lines and 16 columns that we can use in our pro-jects.

Jumper Cable:
The necessary cable to connect Raspberry Pi Pico and other sensors to the Breadboard.

LED:
The necessary cable to connect Raspberry Pi Pico and other sensors to the Breadbo-ard.

Buzzer:
Buzzer; It is a type of auditory warning device that works based on mechanical, elect-romechanical or piezoelectric principles. Buzzers, which have a lot of usage areas, ge-nerally work with the piezoelectric principle. Buzzers can be used in functions such as alarm, timer, confirmation and response warning depending on their usage areas.

HC-SR04 Ultrasonic Sensor:
Buzzer; It is a type of auditory warning device that works based on mechanical, electro-mechanical or piezoelectric principles. Buzzers, which have a lot of usage areas, generally work with the piezoelectric principle. Buzzers can be used in functions such as alarm, ti-mer, confirmation and response warning depending on their usage areas.

01

Introducing Raspberry Pi Pico

robotistan

BLOG

maker.robotistan.com

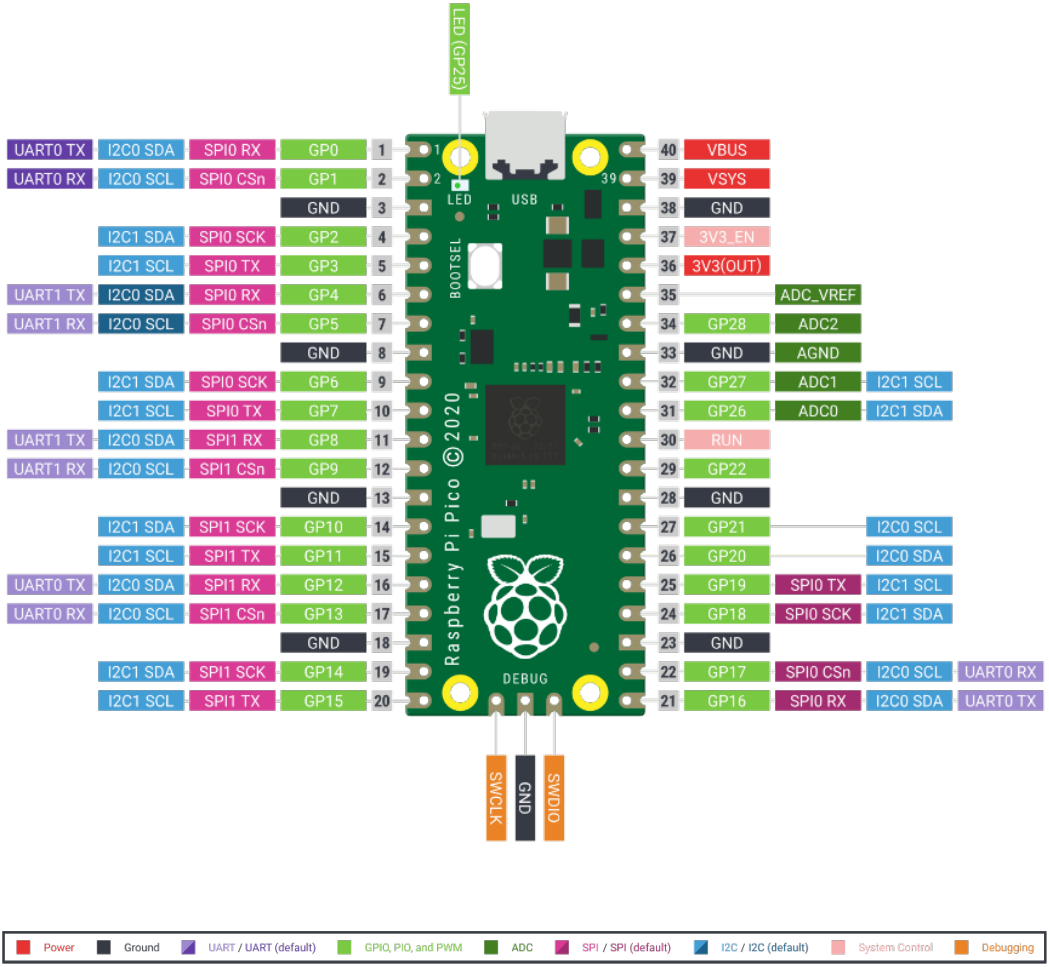
FORUM

forum.robotistan.com

YouTube

youtube.com/robotistan

Raspberry Pi Pico that was released by Raspberry Pi Foundation is a microcontroller that might use in our Embedded systems projects, prototyping or to the Micropython presented to us for both elect-ronic and in learning coding. Raspberry Pi Pico, which is quite powerful compared to Arduino Nano, stands out with its 32-bit Arm Cortex M0 + 133 MHz processor. Thanks to the temperature sensor on it, it will be very easy for us to make projects such as a weather monitor.



02

Installing Thonny IDE

robotistan

BLOG

maker.robotistan.com

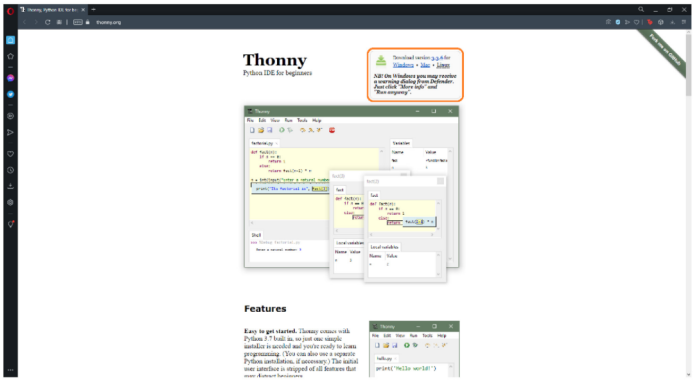
FORUM

forum.robotistan.com

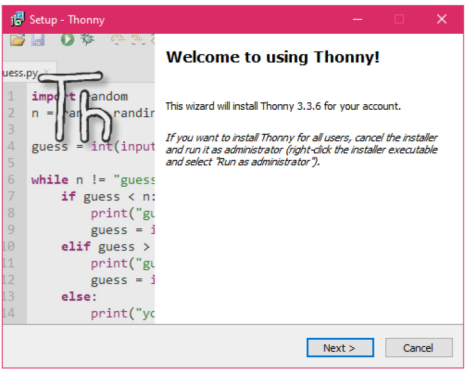
YouTube

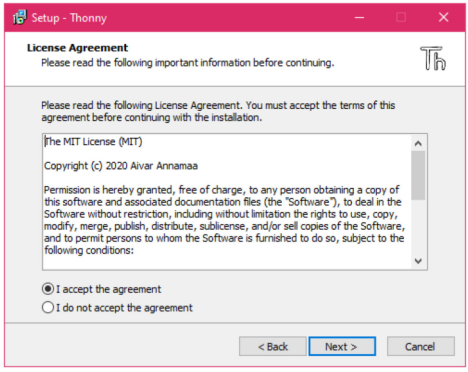
youtube.com/robotistan

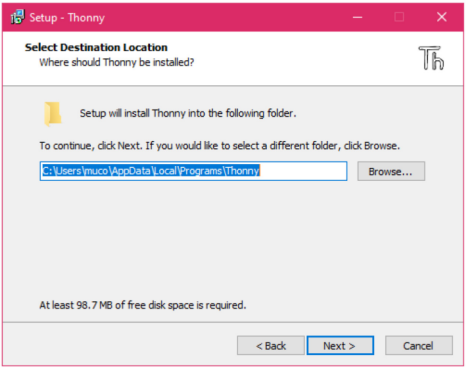
Firstly, You should download Thonny Ide which right version for your operating system on link **https://thonny.org**. In this blog, you will see the version for Windows.

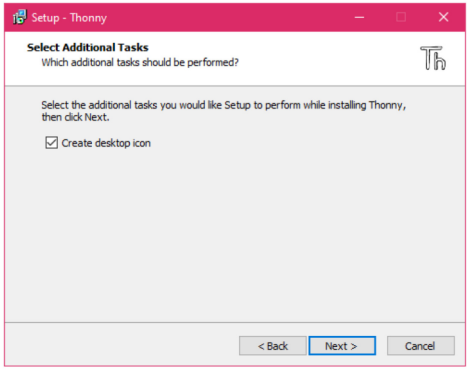


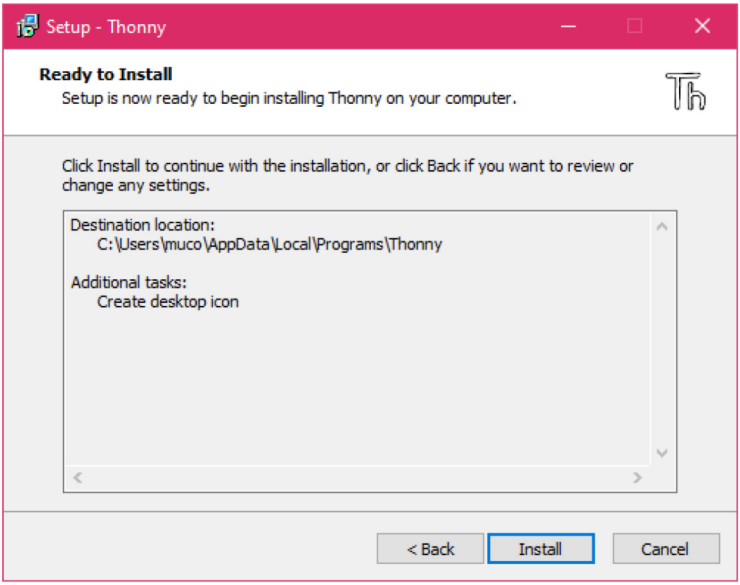
You run the installation file you downloaded. Follow the these screenshots to complete the installa-tion process. You just need to click "Next".

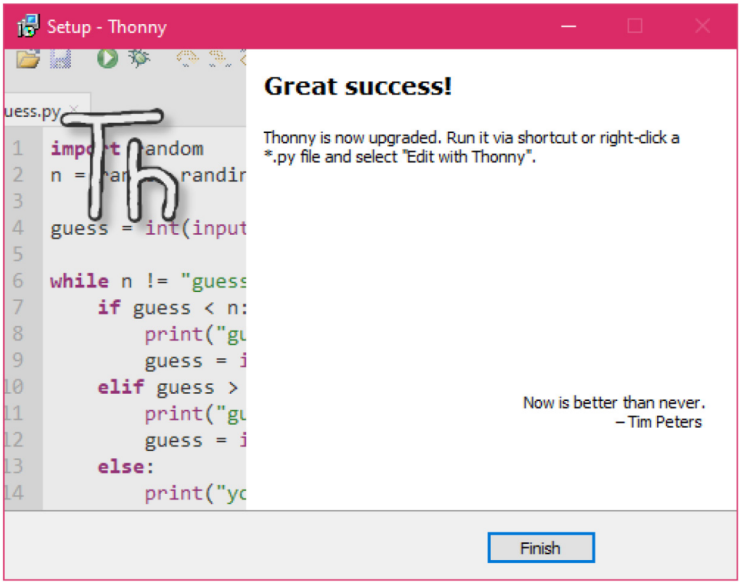




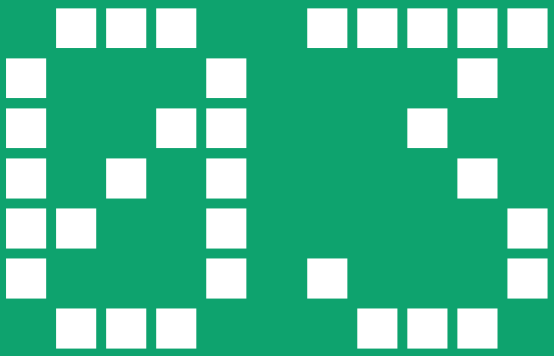








Lastly, You click "Finish"



Installing Micropython Firmware on Raspberry Pi Pico

robotistan

BLOG

maker.robotistan.com

FORUM

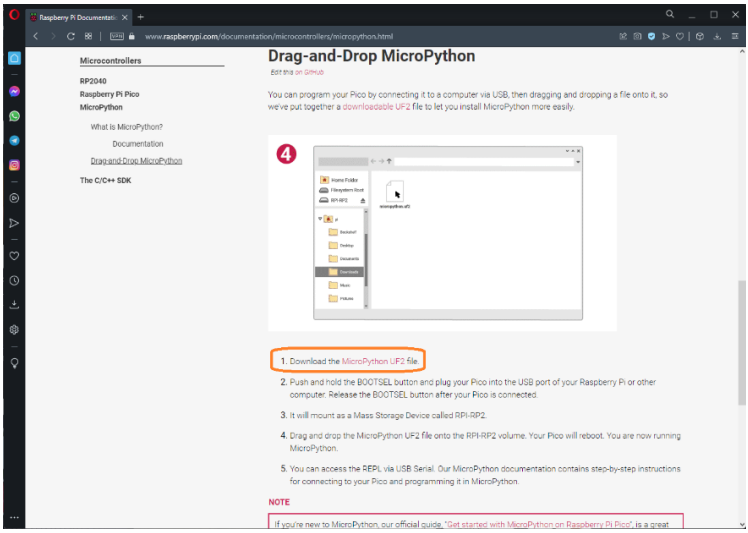
forum.robotistan.com

YouTube

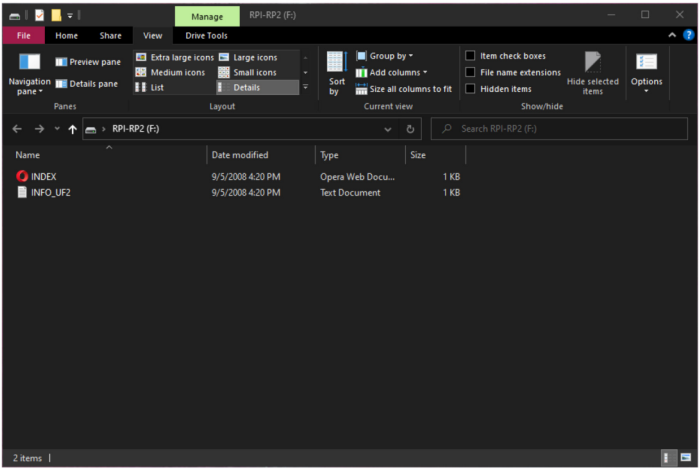
youtube.com/robotistan

Raspberry Pi Pico is programmable in two programming language that MicroPython and "C/C++". You need to download the suitable "firmware" file for programming your Pico.
1- Just follow the instructions to download.
https://www.raspberrypi.com/documentation/mic-rocontrollers/micropython.html

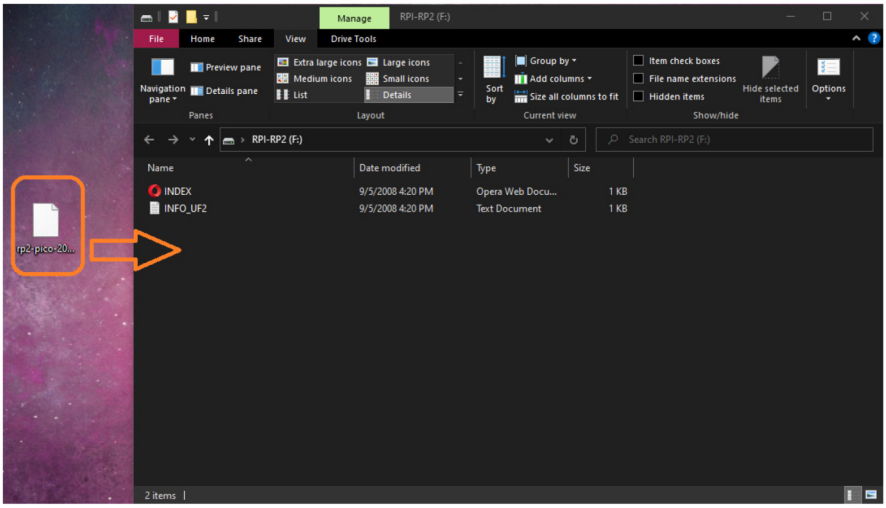
2- You click "Download UF2 file" and download the "firmware"



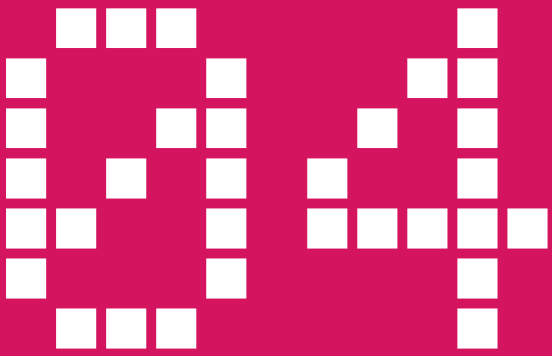
3- Connect to our computer by holding down the "BOOTSEL" button on your Pico. You will see the folder when you connected.



4- Copy the "*.UF2" file that downloaded on that folder.



When the copying process is finished, the folder will close automatically.



Introducing Thonny IDE Interface

robotistan

BLOG

maker.robotistan.com

FORUM

forum.robotistan.com

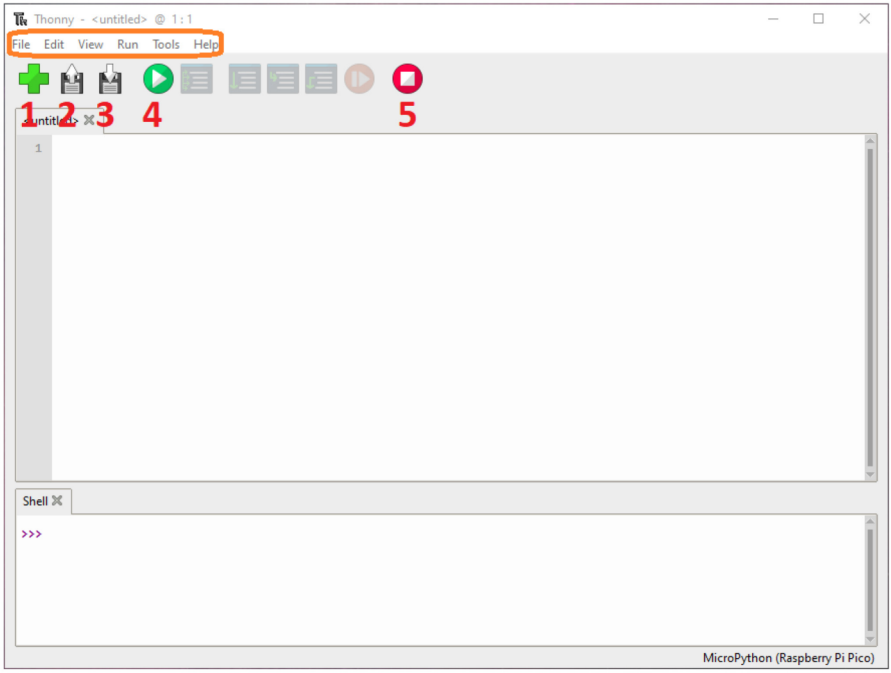
YouTube

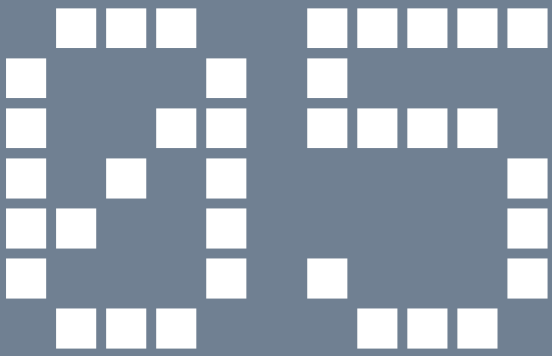
youtube.com/robotistan

Unlike Arduino, Raspberry Pi Pico has its own memory ,and library or scripts what you want can be uploaded into the this memory with Thonny IDE. There are some useful things in Thonny IDE When you open the Thonny IDE, an interface like in the photo will welcome you. The functions of the tabs at the top, the section I have included in the orange box, respectively;

- Files: On this tab, you can save scripts what you wrote, open the scripts you have written or new Project file
- Edit: on this tab you can undo, forward, copy or select all in scripts what you have written
- View: On this tab ,you can customize the interface
- Run: on this tab, you can run code you have written, choose Interpreter for your Rasp-berry Pi Pico or operate like Debug on code
- Tools: On this tab, you can manage packets or edit Thonny settings
- Help: On this tab, you can get information about Thonny version or view help content regarding your issue.

1. You can open a new file with this icon
2. You can open scripts you have saved with this icon
3. You can save the script with this icon
4. You can run the script with this icon abi bu ne salak mı adamlar?
5. You can stop the script with this icon





Blink Internal LED

robotistan

BLOG

maker.robotistan.com

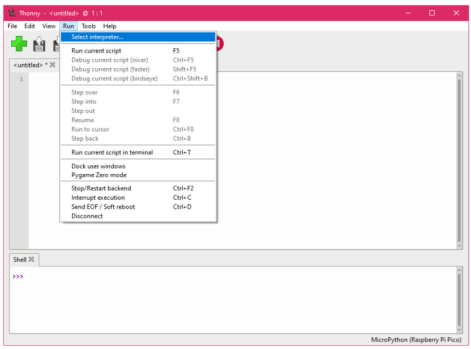
FORUM

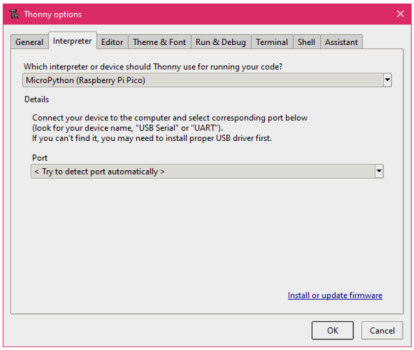
forum.robotistan.com

YouTube

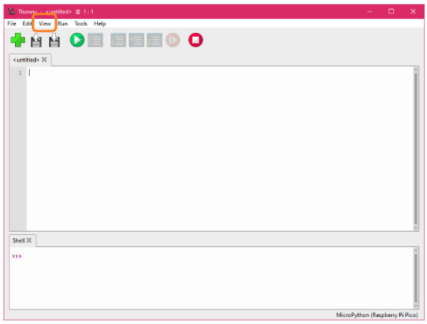
youtube.com/robotistan

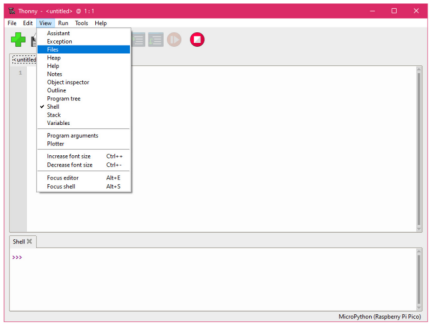
You adjust the Interpreter Settings on Thonny IDE for using your Raspberry Pi Pico. Click the the run tab at the top, then click Select Interpreter. Select "MicroPython (Raspberry Pi Pico)" as the "Interpreter" and "Try to detect port automatically" as the port from the window that opens.



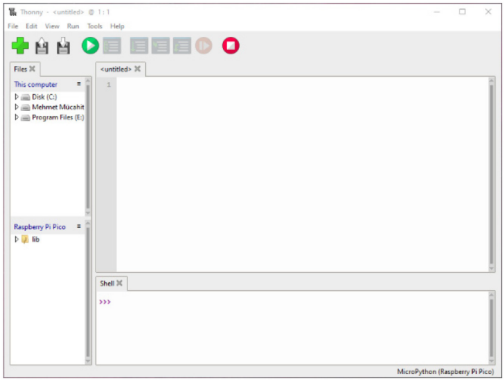


Add "Files" tab that will contribute on the interface.Click the the"View" tab at the top and select "Files".





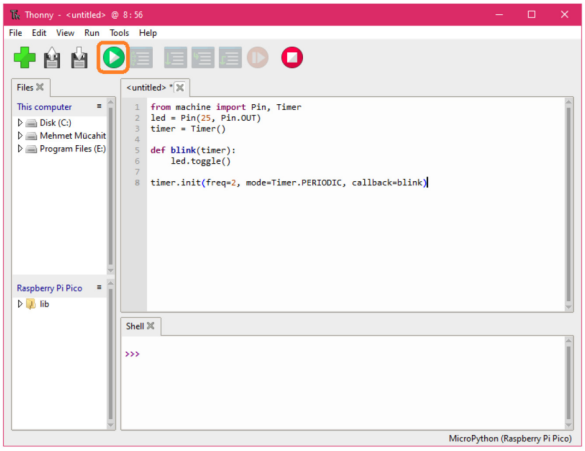
After that,you will see a tab like these screenshots..



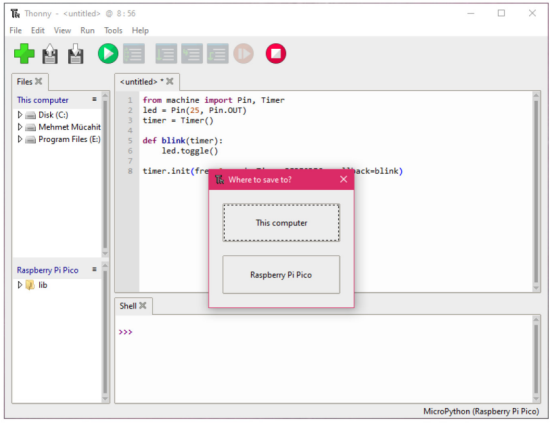
Type the following code into Thonny IDE this code provides the Internal LED to blink every half second When you examine this code, you wrote the code "from machine import Pin, Timer" so that be able to access the hardware on the Pico. In line 2, you stated that the internal LED on the Pico is connected to pin 25. In line 3, you started timer .In line 5,you defined a blink function. This is a simple function that makes the LED light. In line 8, you ran the timer. We specified the frequency of the pro-cess with "freq" in parentheses, the mode of the timer with the "mode" and the function we would call with the "callback".

```
from machine import Pin, Timer
led = Pin(25, Pin.OUT)
timer = Timer()

def blink(timer):
    led.toggle()

timer.init(freq=2, mode=Timer.PERIODIC, callback=blink)
```

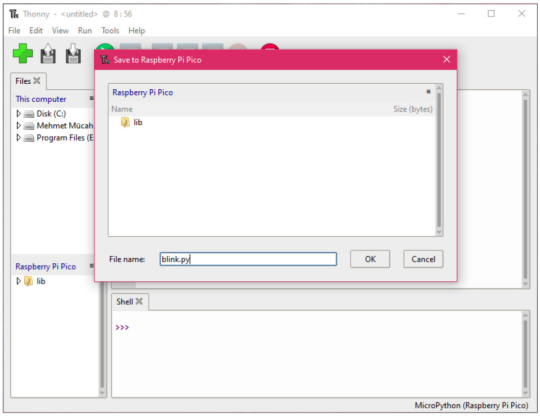
Upload this code to your Pico. Let's click on the green "Run" icon at the top.



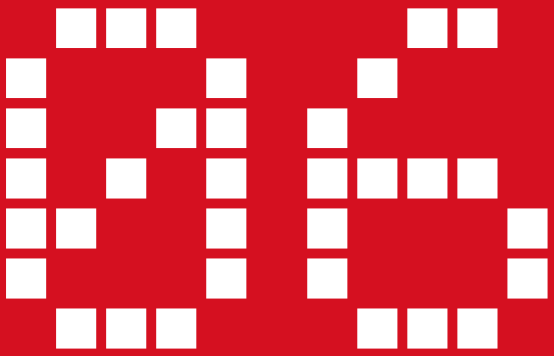
It asks where you want to save the code. If save the code to the computer and run it, the code will only run when the Pico is plugged into the computer. For the code to always run. You need to click on the "Raspberry Pi Pico" option and save it to your Pico.



After clicking on the "Raspberry Pi Pico" text, a window like in the photo will open. In this window, it asks us to which location you want to save the code you wrote in Pico. You can save it directly in Pico. For this, you need to write a name and file extension to save the code youhave written. Since you are working with MicroPython, your file extensions will always be "* .py". I typed "blink.py" as the filename. You can also type any file name you want. After typing the file name, you can save and run the code by clicking the "OK" button.



When the code runs, the internal LED on your Pico will blink and flash at half-second intervals.



LED Control with Button

robotistan

BLOG

maker.robotistan.com

FORUM

forum.robotistan.com

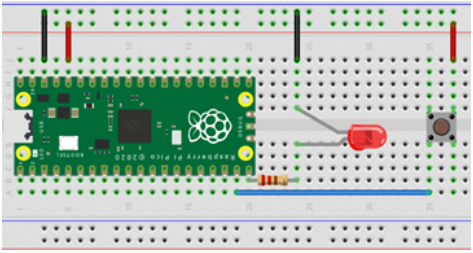
YouTube

youtube.com/robotistan

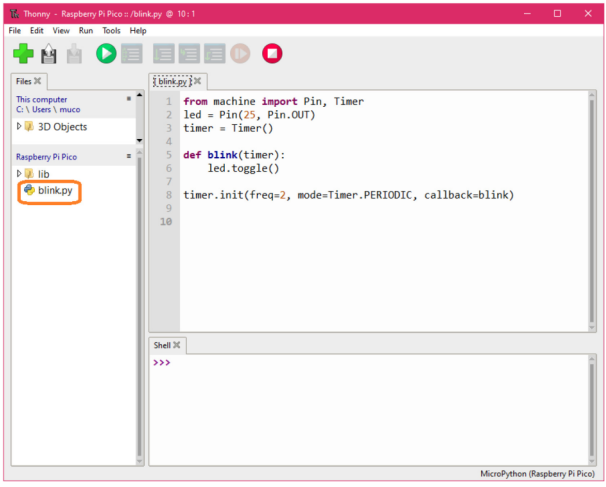
Once you lit LEDs, but this time, you need to realize this project in order to warm up our Pico's hardware and breadboard usage.

Necessary Materials:
- Raspberry Pi Pico
- Breadboard
- Push Button
- LED
- Male – Male Jumper
- 220 Ω or 330 Ω Resistor

When you look at the pin diagram of your Pico, the 38th pin is "GND", ie the ground pin, the 36th pin is the "3V3 (OUT)" pin. You will use these two pins frequently in your projects.
First, build the circuit on the breadboard according to the diagram below.



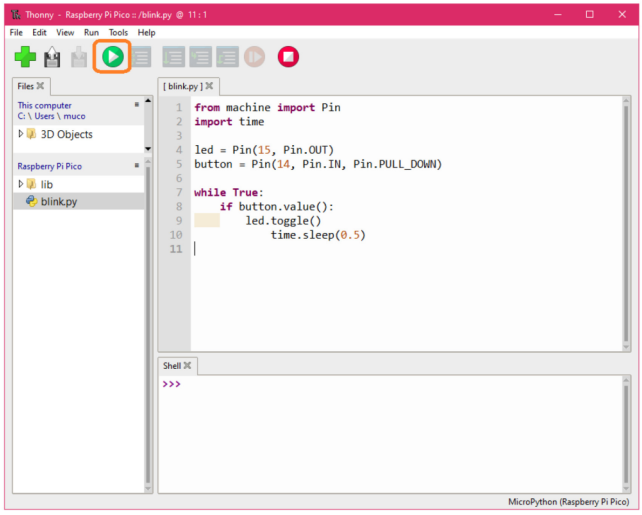
Then open your Thonny IDE and edit the "blink" code you have written in the previous project. Open it by double clicking on the "blink.py" file on the left side of Thonny IDE. The codes you have written in the previous project will be opened. Now you will delete these codes completely and write new code.



When you examine the codes below, you defined the hardware of Pi-co with the code "from mac-hine import Pin" as in the previous project and added the "time" library. With the code "led = Pin (15, Pin.OUT)", you assigned the 15 nu-mara pin as the pin you connected to the LED. With the code "but-ton = Pin (14, Pin.IN, Pin.PULL_DOWN)", you defined the pin number 14 to which you connect the button as the input pin. Then you created a "while loop". In this cycle, it will change the status of the led according to the value read from the button. For example, if press the button once while the LED is off, it will turn on, if the LED is on, it will turn off when you press the button. When press and hold the button with the code "time.sleep (0.5)", the LED will flash and flash at half-second intervals.

```
from machine import Pin
import time

led = Pin(15, Pin.OUT)
button = Pin(14, Pin.IN, Pin.PULL_DOWN)

while True:
    if button.value():
            led.toggle()
                time.sleep(0.5)
```

After writing the codes, press the "Run" icon at the top and save your codes to your Pico and run them.



Now you will be able to turn the LED on and off by pressing the button. Move on to the next project.

07

Temperature Measurement with Pico

robotistan

BLOG

maker.robotistan.com

FORUM

forum.robotistan.com

YouTube

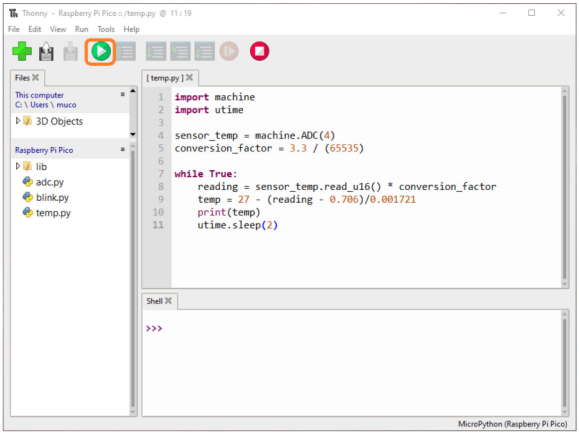
youtube.com/robotistan

As I mentioned at the beginning, Raspberry Pi Pico has an internal temperature sensor. So why not take a temperature measurement?

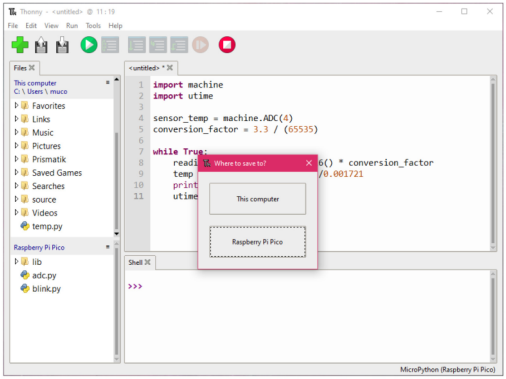
First, open the Thonny IDE and open a new file and write the following codes. Now, when you examine the codes, you have added the necessary "machine" and "utime" libraries as usual. Since made analog readings with the temperature sensor, you defined that it was connected to pin number 4 with the code "sensor_temp = machine. ADC (4)", then you converted the 16-bit data from the sen-sor into voltage data that could be meaningful for you with the code "conversion_factor". Since the Pico pin gives 3.3 volts, you divided 3.3 volts by 216 - 1 = 65535. Now, when you examine the While loop, read the data from the sensor. In the code on line 8, multiply the data from the temperature sensor with the "conversion_factor" you just wrote, and write the temperature data in terms of vol-tage. You convert the data you have obtained to "Celsius" in the 9th line. Finally, print the tempera-ture data to Shell with the code "print (temp)".

```
 1 import machine
 2 import utime
 3
 4 sensor_temp = machine.ADC(4)
 5 conversion_factor = 3.3 / (65535)
 6
 7 while True:
 8     reading = sensor_temp.read_u16() * conversion_factor
 9     temp = 27 - (reading - 0.706)/0.001721
10     print(temp)
11     utime.sleep(2)
```

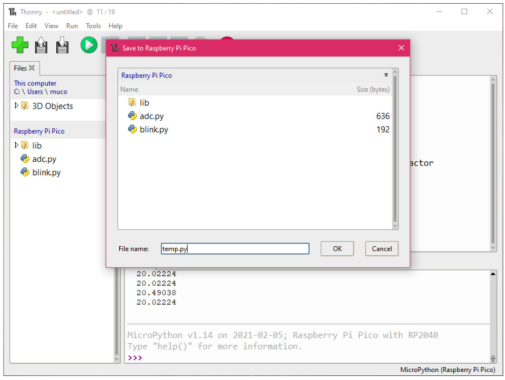
Now save these codes to your Pico and run them, then press the "Run" icon at the top.



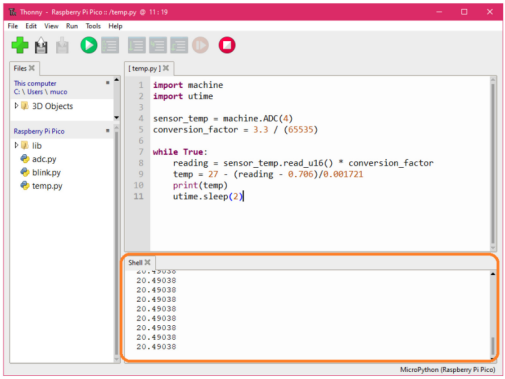
It asks where you want to save the code. For the code to always run, you need to click on the "Raspberry Pi Pico" option and save it to our Pico.

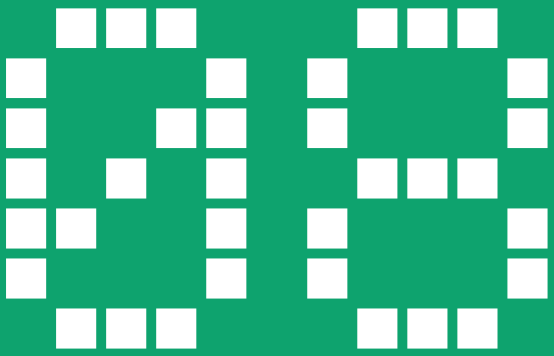


Write "temp.py" as the file name, click the "OK" button and now you can read the temperature data from Shell.



In the Shell window at the bottom, you can see the temperature data from your Pico every 2 seconds.





Weather Monitor

robotistan

BLOG

maker.robotistan.com

FORUM

forum.robotistan.com

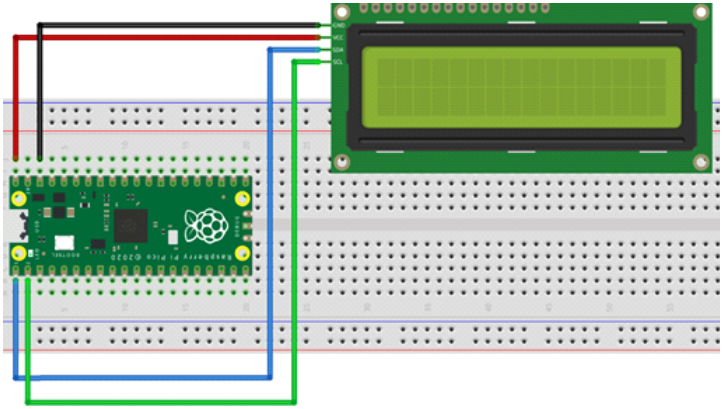
YouTube

youtube.com/robotistan

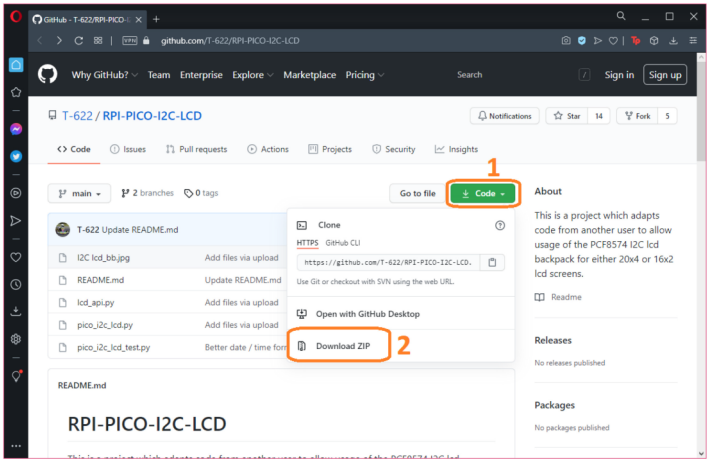
In this project, you will make a weather monitor using a 2X16 LCD screen. In this project, you will learn how to add a library and automatically save the temperature data you measure in a text file.

Necessary materials:
- Raspberry Pi Pico
- Breadboard
- 2X16 LCD Screen
- Male – Female Jumper

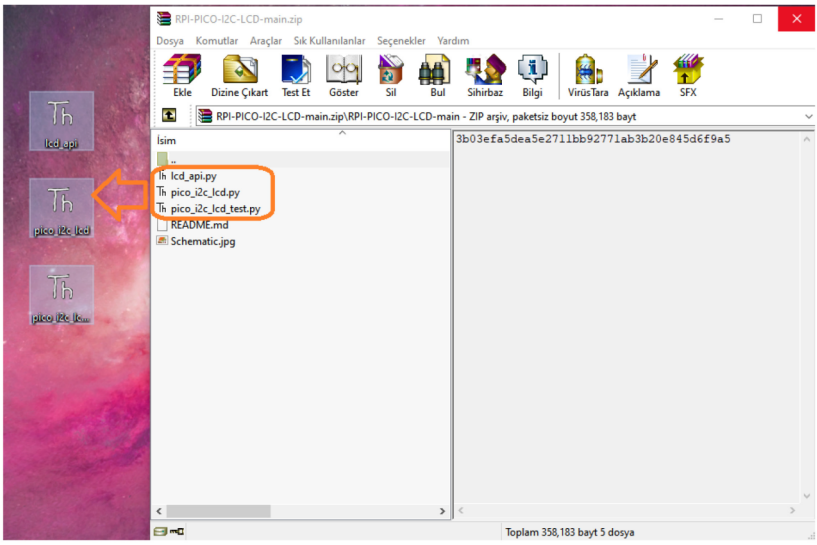
First, build your own circuit on the breadboard according to the circuit below.



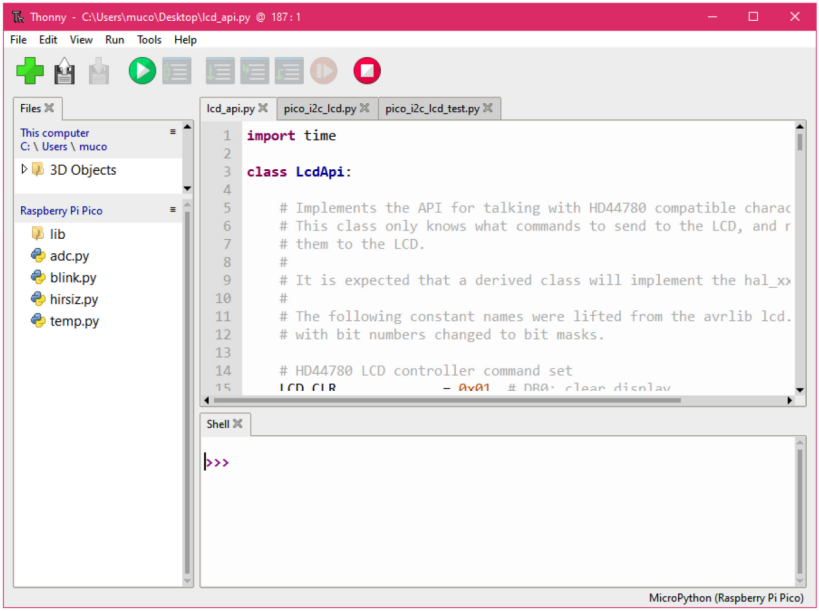
Now download the necessary library to use our LCD screen from **https://github.com/T-622/RPI-PICO-I2C-LCD.** When open the link, click the green button that says "Code" and then click the "Download ZIP" button to download the library.



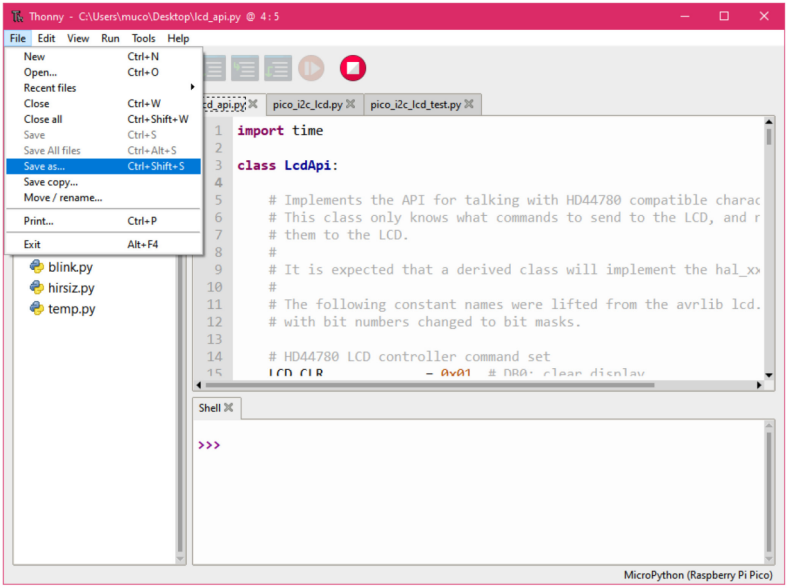
Open the downloaded "* .zip" file and extract the "lcd_api.py", "pico_i2c_lcd.py", "pico_i2c_lcd_test.py" files to the desktop.



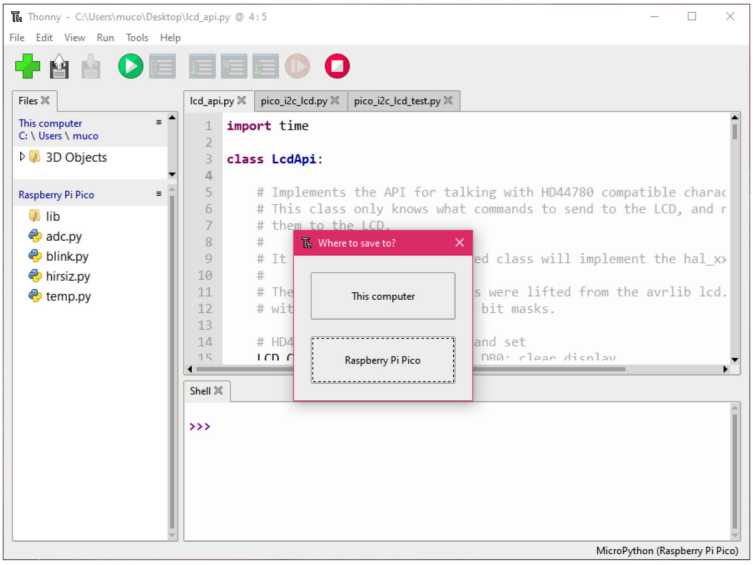
Open them in Thonny IDE by double clicking on these files that you extract to the desktop.



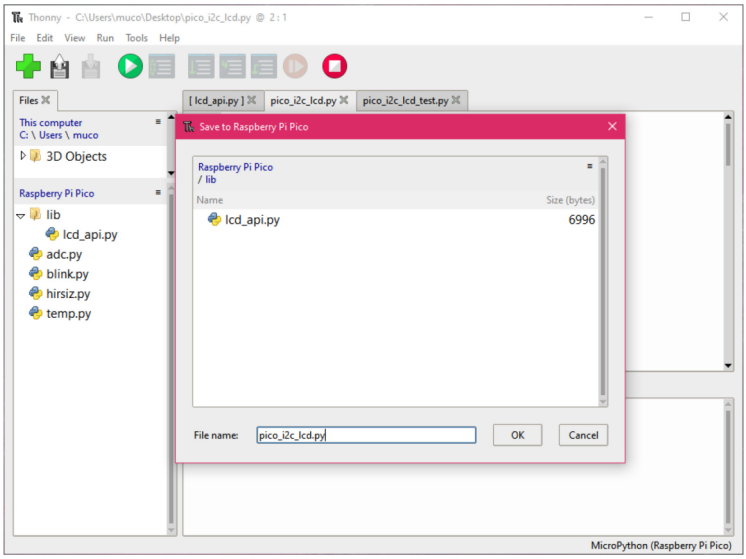
Now save these files to the "lib" folder in your Pico in order, click on the "Files" tab on the top and then click on "Save as ...".

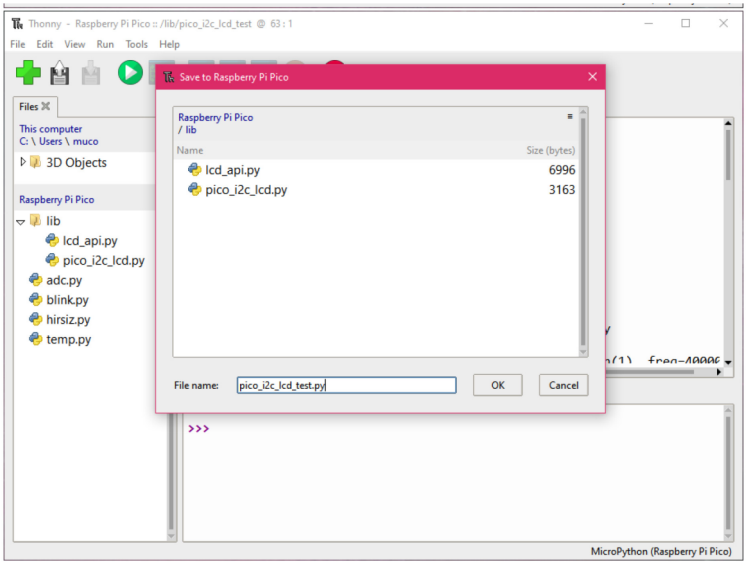


It asks where you want to save the library file. You will choose the Raspberry Pi Pico.



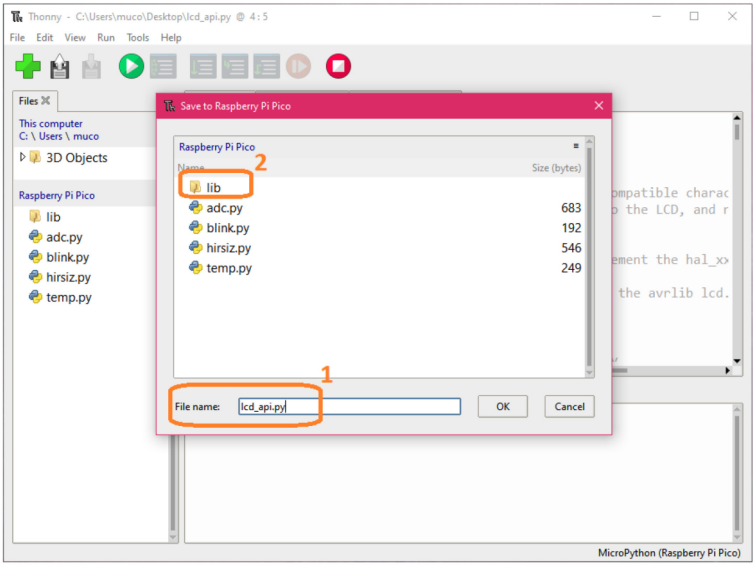
Do the same for the other two files.



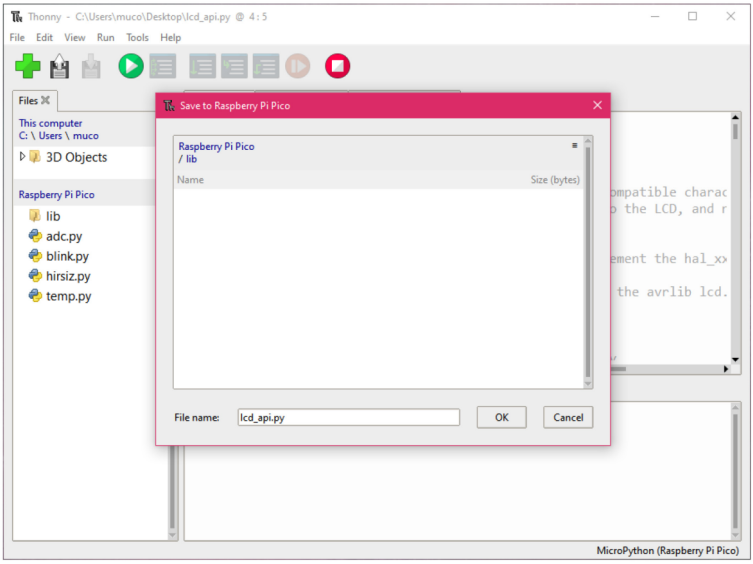


Now that you have added libraries, you can open a new file by clicking the "Plus" icon on the top of a new Thonny IDE and write the following codes. When you examine the codes, first added the libra-ries as in other projects, then made the settings of our 2X16 LCD screen and created a new text docu-ment named "save" with the code "file = open (" record.txt "," w ").

In this part, you have to choose the location where want to save the file, then type the file name first. After that double click on the "lib" folder.

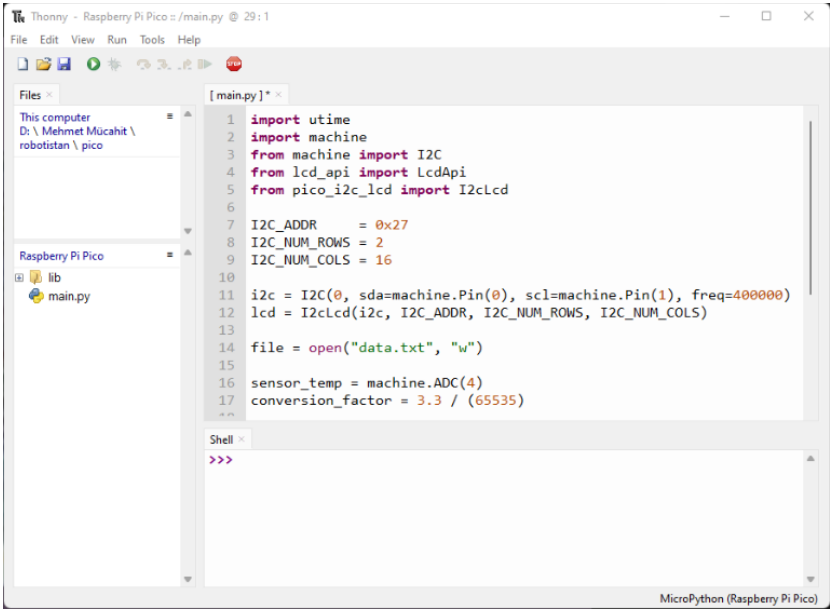


Then click the "OK" button to save it to the folder.

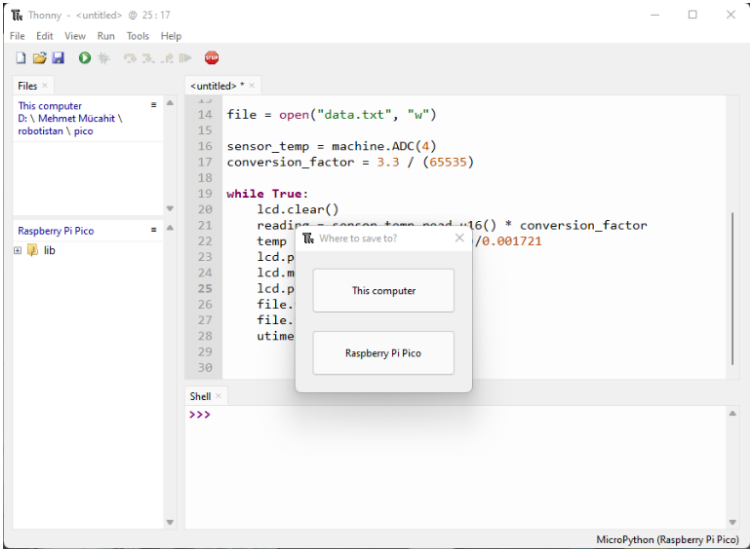


```
import utime
import machine
from machine import I2C
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd

I2C_ADDR     = 0x27
I2C_NUM_ROWS = 2
I2C_NUM_COLS = 16

i2c = I2C(0, sda=machine.Pin(0), scl=machine.Pin(1), freq=400000)
lcd = I2cLcd(i2c, I2C_ADDR, I2C_NUM_ROWS, I2C_NUM_COLS)

dosya = open("data.txt", "w")

sensor_temp = machine.ADC(4)
conversion_factor = 3.3 / (65535)

while True:
    lcd.clear()
    reading = sensor_temp.read_u16() * conversion_factor
    temp = 27 - (reading - 0.706)/0.001721
    lcd.putstr("Temp: ")
    lcd.move_to(0,1)
    lcd.putstr(str(temp))
    dosya.write(str(temp) + "\n")
    dosya.flush()
    utime.sleep(2)
```

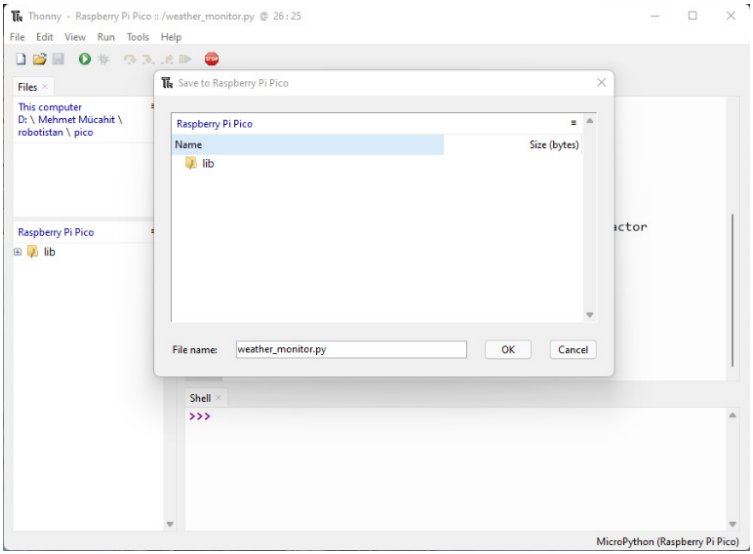
Now save these codes to your Pico and run them, then press the "Run" icon at the top.



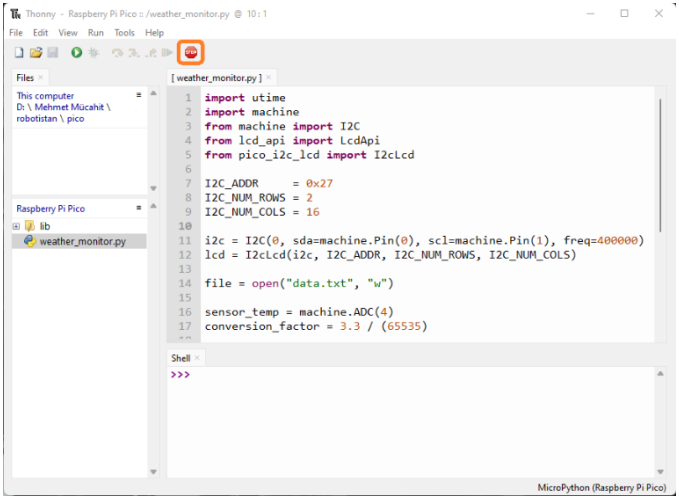
It asks where you want to save the code. For the code to always run, you need to click on the "Raspberry Pi Pico" option and save it to our Pico.



Type "weather_monitor.py" as the file name and click the "OK" button. Now, you will be able to see the temperature data on your LCD screen and at the same time this data will be recorded in a text document.

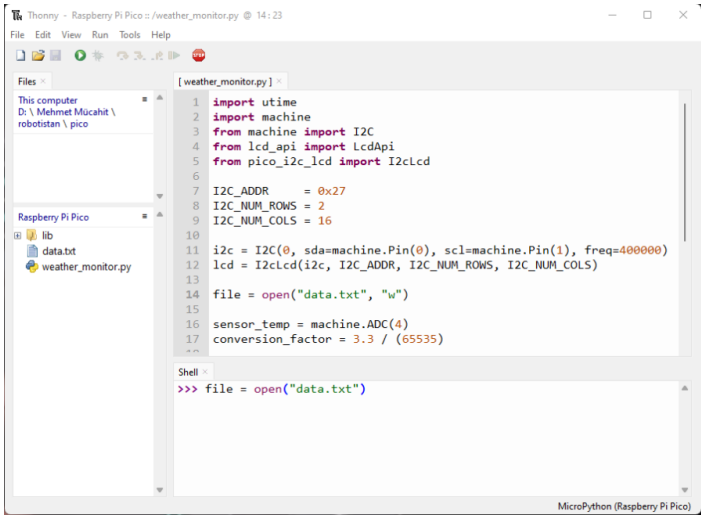


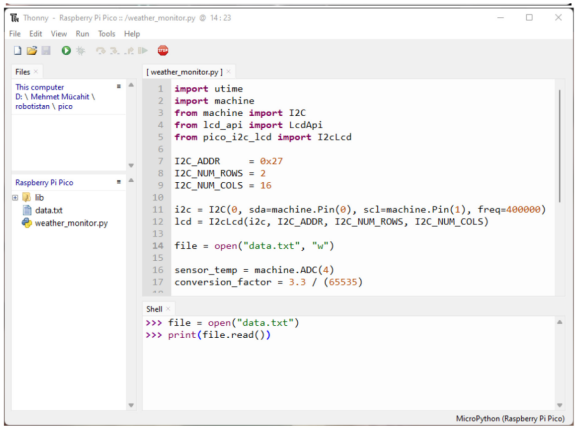
Now wait for some data to collect, then click the "Stop" icon at the top to stop the code from running.



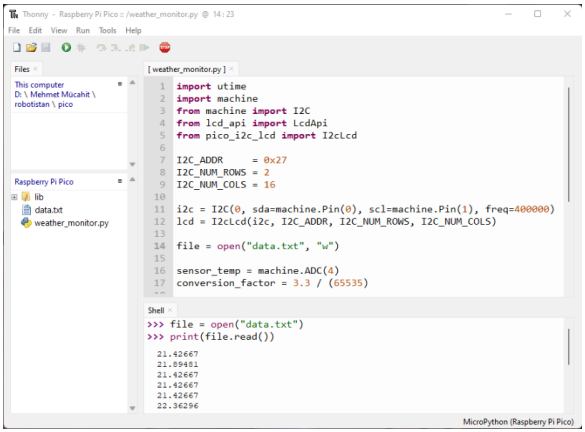
Write the following codes in the Shell section at the bottom so that you can see the data have saved.

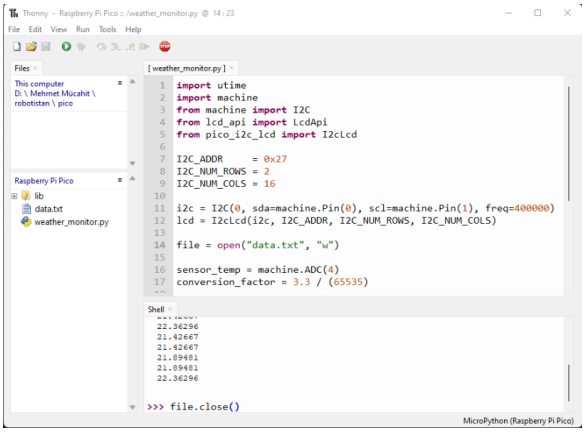
```
file = open("data.txt")
print(file.read())
file.close()
```

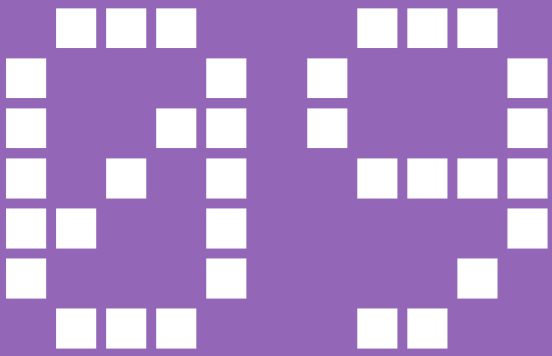




After typing the "print (file.read ())" code, you can see the saved data as in the photo below.







Burglar Alarm

robotistan

BLOG

maker.robotistan.com

FORUM

forum.robotistan.com

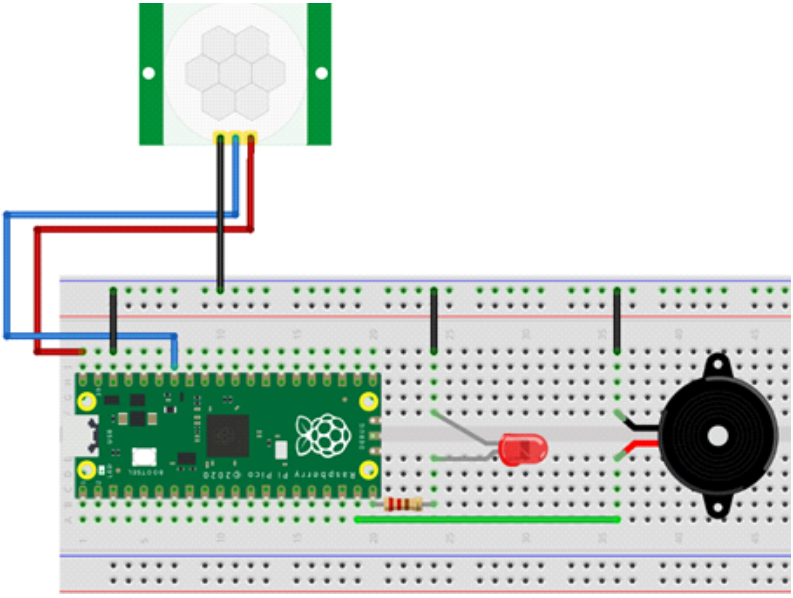
YouTube

youtube.com/robotistan

In this project, you will make a burglar alarm using PIR sensor, LED and buzzer.

Necessary materials:
- Raspberry Pi Pico
- Breadboard
- PIR Sensor
- Buzzer
- LED
- Male – Male Jumper
- Male – Female Jumper
- 220 Ω or 330 Ω Resistor

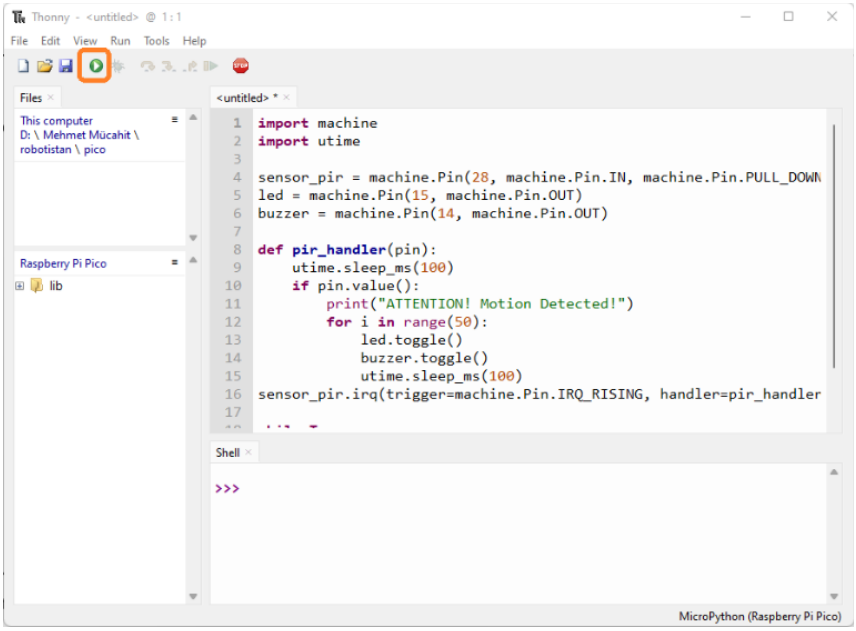
First, build your circuit according to the diagram below.



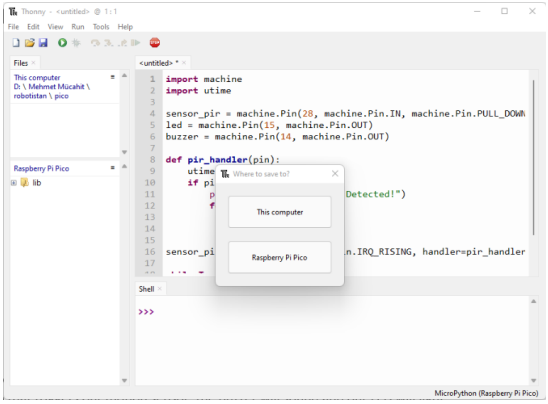
Now open a new file in Thonny IDE and write the following codes. . When you examine the codes, you first added the libraries and then defined the pins to which you connected the PIR sensor, Buzzer and LED. You created a function with "def pir_handler (pin)". When the motion is detected with the if block in the function, "ATTENTION! Motion Detected! " printing, buzzer and LED are wor-king.

```
import machine
import utime

sensor_pir = machine.Pin(28, machine.Pin.IN, machine.Pin.PULL_DOWN)
led = machine.Pin(15, machine.Pin.OUT)
buzzer = machine.Pin(14, machine.Pin.OUT)

def pir_handler(pin):
    utime.sleep_ms(100)
    if pin.value():
        print("ATTENTION! Motion Detected!")
        for i in range(50):
            led.toggle()
            buzzer.toggle()
            utime.sleep_ms(100)
sensor_pir.irq(trigger=machine.Pin.IRQ_RISING, handler=pir_handler)

while True:
    led.toggle()
    utime.sleep(5)
```

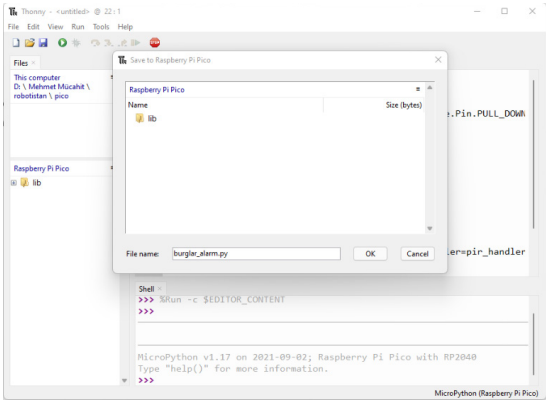
Now save these codes to your Pico and run them, then press the "Run" icon at the top.



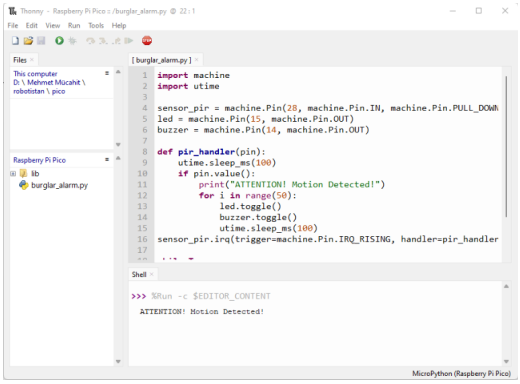
It asks where you want to save the code. For the code to always run, you need to click on the "Raspberry Pi Pico" option and save it to our Pico.



Write "burglar_alarm.py" as the file name and click on the "OK" button and now, in case of any situation that triggers our motion sensor, the buzzer will sound and our LED will light.



In the Shell window at the bottom, when motion is detected, "ATTENTION! Motion Detected! " text will appear.



10

Using of Distance Sensor

robotistan

BLOG

maker.robotistan.com

FORUM

forum.robotistan.com

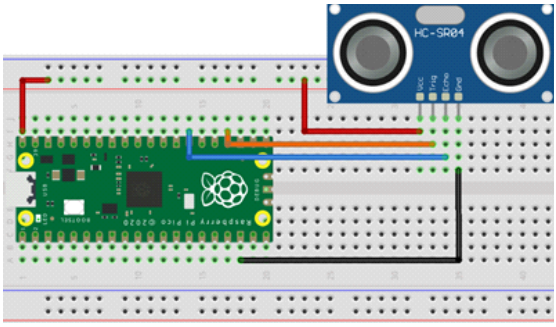
YouTube

youtube.com/robotistan

In this project, you will measure distance with HC-SR04 distance sensor.

Necessary materials:
- Raspberry Pi Pico
- Breadboard
- HC-SR04 Distance Sensor
- Male – Male Jumper

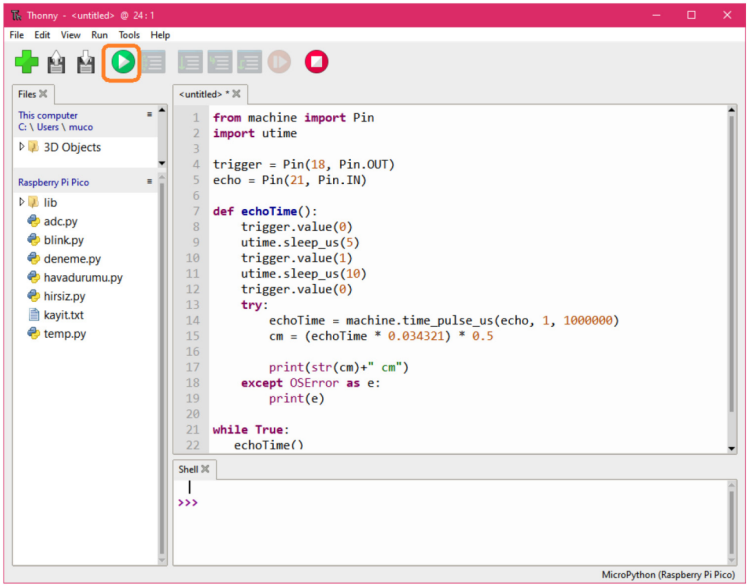
First, create your circuit on the breadboard according to the diagram below.



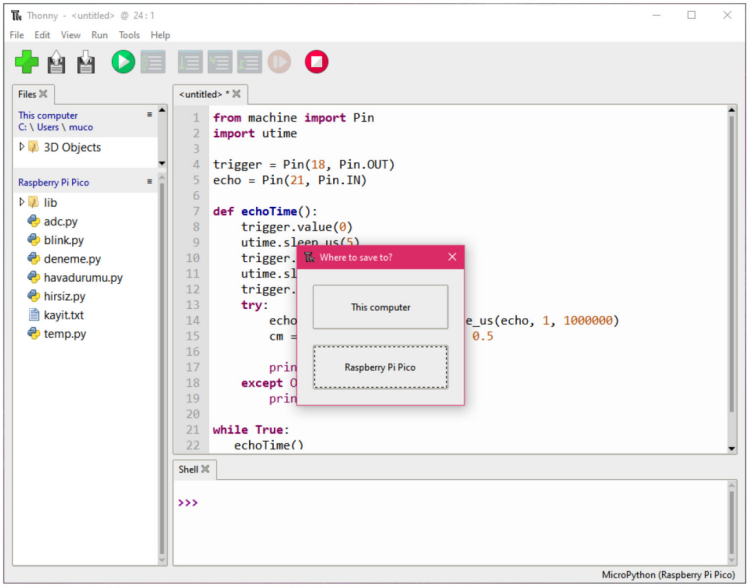
Now open a new file in Thonny IDE and write your code. When you examine the codes, you defined the libraries in the first two lines, and the pins to which the distance sensor is connected. Also wrote the distance measurement function with the function "def echoTime ():". You have made the distance measurement function, which defined with the while loop, run every 5 seconds.

```
from machine import Pin
import utime

trigger = Pin(18, Pin.OUT)
echo = Pin(21, Pin.IN)

def echoTime():
    trigger.value(0)
    utime.sleep_us(5)
    trigger.value(1)
    utime.sleep_us(10)
    trigger.value(0)
    try:
        echoTime = machine.time_pulse_us(echo, 1, 1000000)
        cm = (echoTime * 0.034321) * 0.5

        print(str(cm)+" cm")
    except OSError as e:
        print(e)

while True:
   echoTime()
   utime.sleep(5)
```

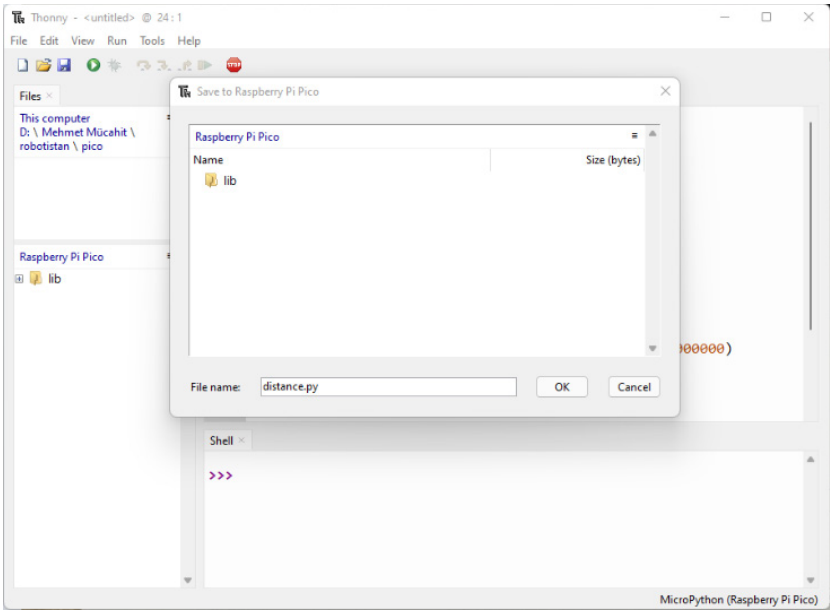
Now save these codes to your Pico and run them, then press the "Run" icon at the top.



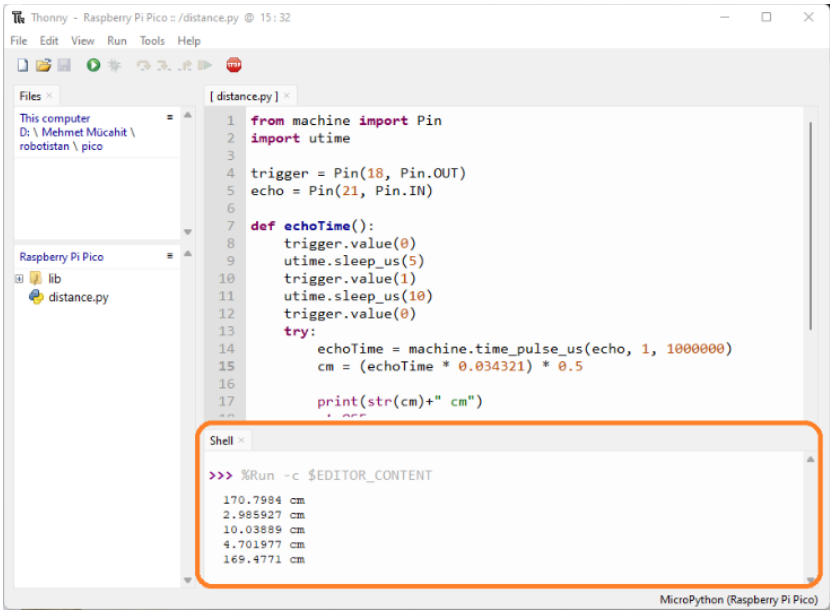
It asks where you want to save the code. For the code to always run, you need to click on the "Raspberry Pi Pico" option and save it to our Pico.



Write "distance.py" as the file name and click the "OK" button.



You can now see the distance in the Shell section of the Thonny IDE.



YouTube

youtube.com/robotistan

FORUM

robotistan

forum.robotistan.com

BLOG

robotistan

maker.robotistan.com

Robotistan Elektronik Ticaret A.Ş.

Mehmet Mücahit KAYA(Content) - Mehmet AKÇALI - Yasin TAŞCIOĞLU (Editor) - Mehmet Nasır KARAER (Graphic)
info@robotistan.com - www.robotistan.com
Tel: 0850 766 0 425