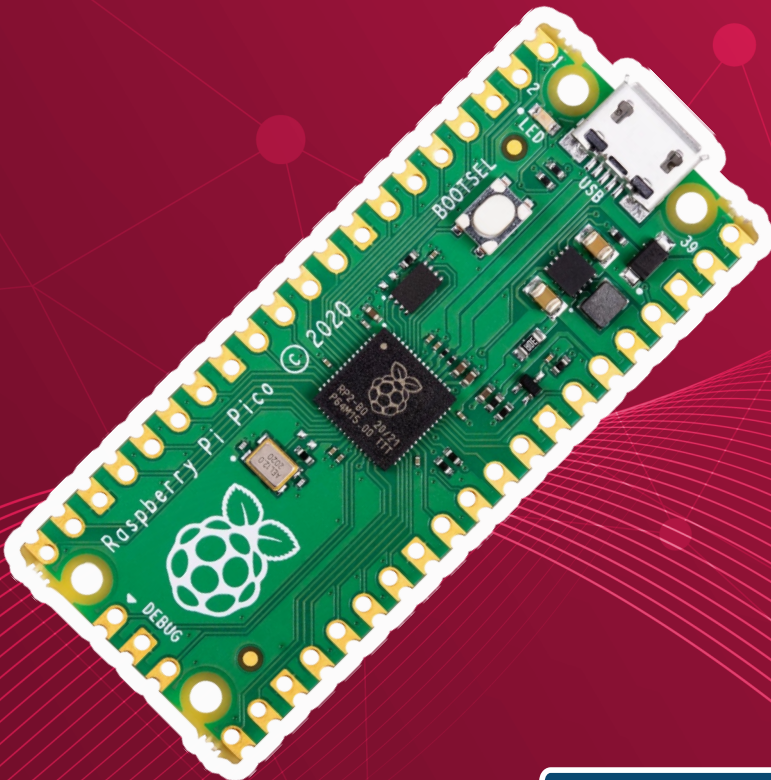


## Raspberry Pi Pico Project Book



robotistan





Welcome to the World of Electronics and Coding. Now that you have opened this book, you are eager to swim 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 written 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](mailto:info@robotistan.com)"

**Robotistan Team**

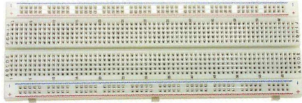
# Contents

Introducing The Kit.....	4
Introducing Raspberry Pi Pico.....	6
Instaling Thonny IDE.....	8
Installing Micropython Firmware on Raspberry Pi Pico.....	11
Introducing Thonny IDE Interface.....	14
Blink Internal LED.....	16
LED Control with Button.....	20
Temperature Measurement with Pico.....	23
Weather Monitor.....	26
Burglar Alarm.....	36
Using of Distance Sensor.....	40



### 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 before transferring them to printed circuit boards or perforated plates.



### HC-SR501 PIR Sensor

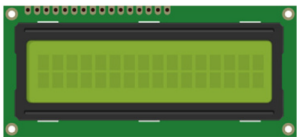
PIR sensors are sensors used to detect the movement of living things in an environment. This tiny sensor is a module that can be used with many microcontroller platforms such as Raspberry Pi, Arduino, ESP32, which you can use comfortably in various electronics, robotics and hobby applications.

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.



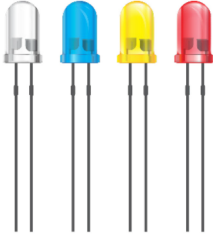
### 2X16 LCD Screen:

It is a screen consisting of 2 lines and 16 columns that we can use in our projects.



### Jumper Cable:

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



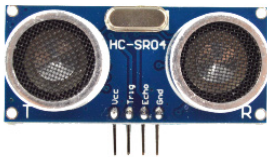
### LED:

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



### Buzzer:

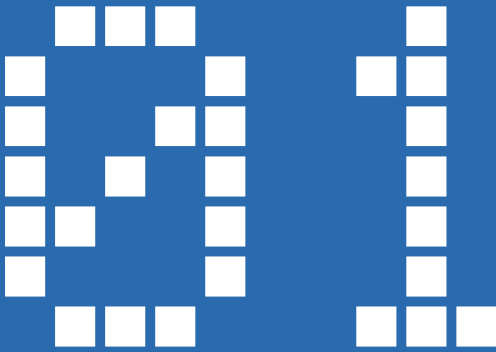
Buzzer; It is a type of auditory warning device that works based on mechanical, electro-mechanical or piezoelectric principles. Buzzers, which have a lot of usage areas, generally work with the piezoelectric principle. Buzzers can be used in functions such as alarm, timer, confirmation and response warning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





# Introducing Raspberry Pi Pico



BLOG



[maker.robotistan.com](https://maker.robotistan.com)

FORUM



[forum.robotistan.com](https://forum.robotistan.com)



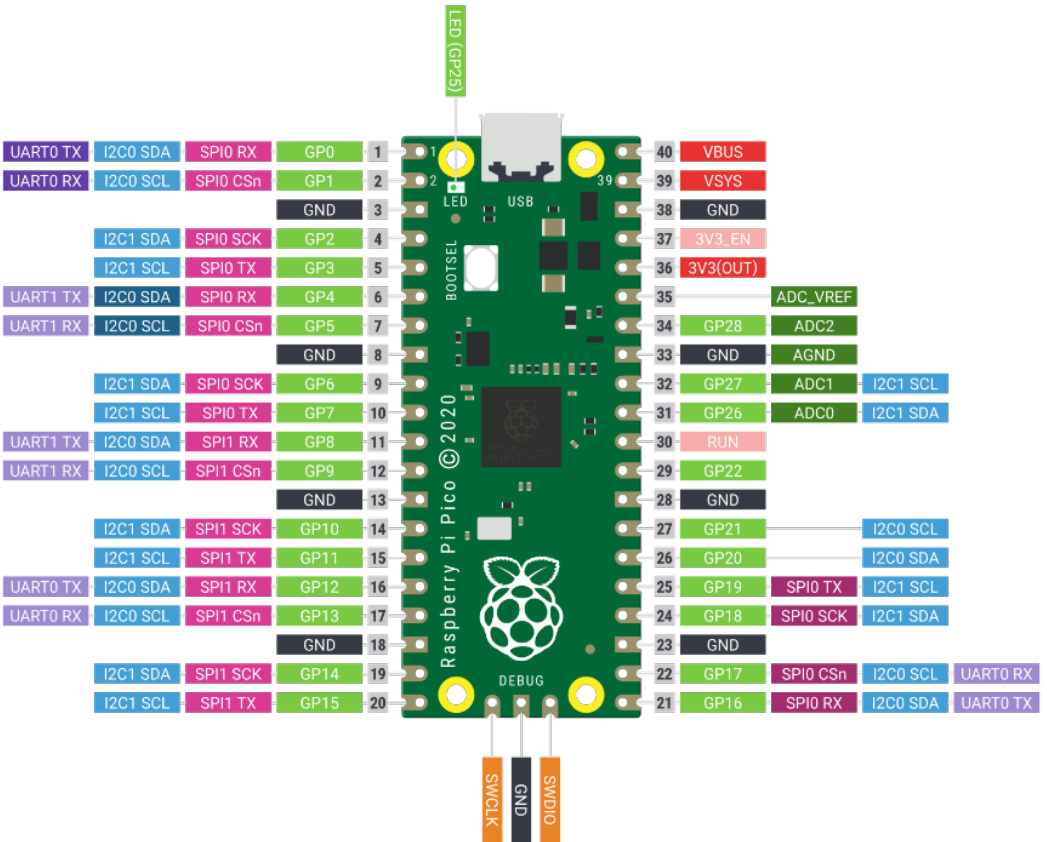
YouTube

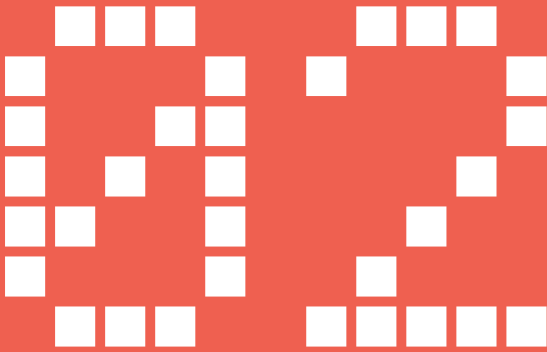


[youtube.com/robotistan](https://youtube.com/robotistan)

# Introducing Raspberry Pi Pico

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.





# Installing Thonny IDE

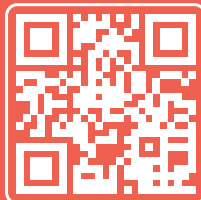


BLOG



[maker.robotistan.com](https://maker.robotistan.com)

FORUM



[forum.robotistan.com](https://forum.robotistan.com)



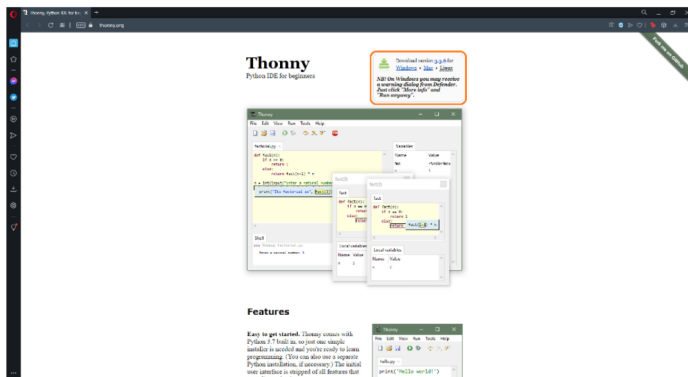
YouTube



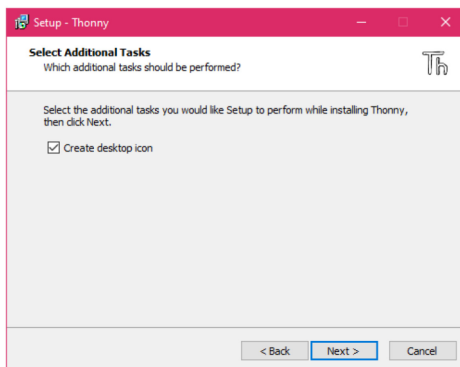
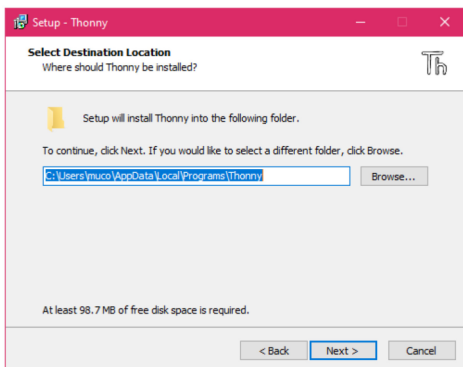
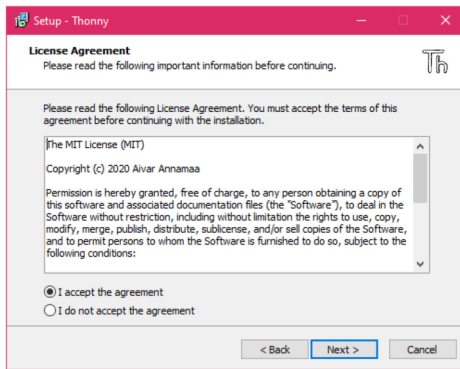
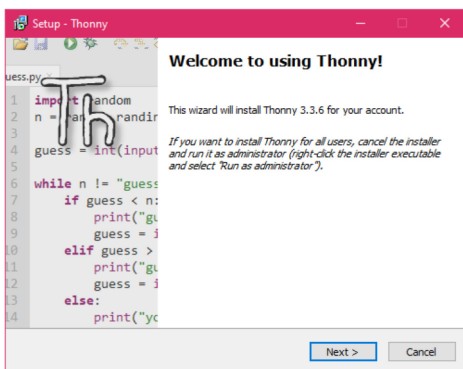
[youtube.com/robotistan](https://youtube.com/robotistan)

# Installing Thonny IDE

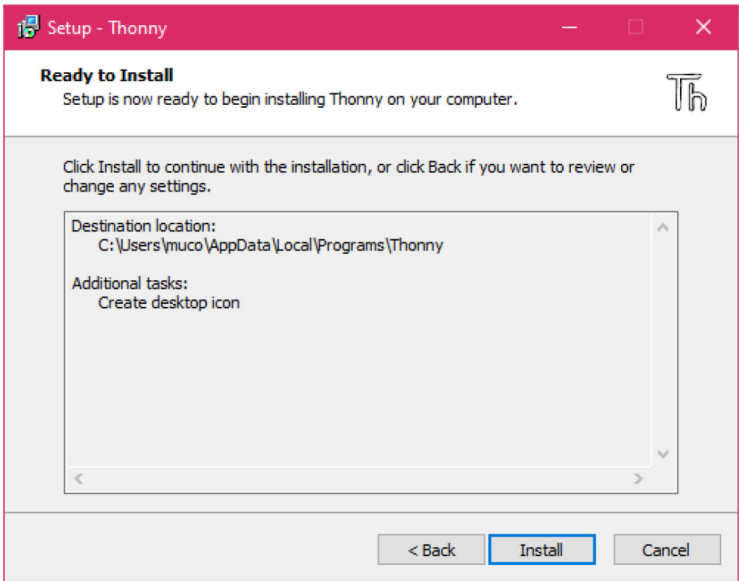
Firstly, You should download Thonny Ide which right version for your operating system on link <https://thonny.org>. In this blog, you will see the version for Windows.



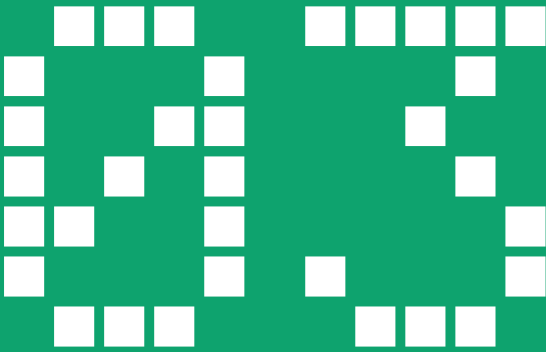
You run the installation file you downloaded. Follow the these screenshots to complete the installation process. You just need to click "Next".







Lastly, You click "Finish"



# Installing Micropython Firmware on Raspberry Pi Pico

robotistan



BLOG



[maker.robotistan.com](https://maker.robotistan.com)

FORUM



[forum.robotistan.com](https://forum.robotistan.com)



YouTube



[youtube.com/robotistan](https://youtube.com/robotistan)

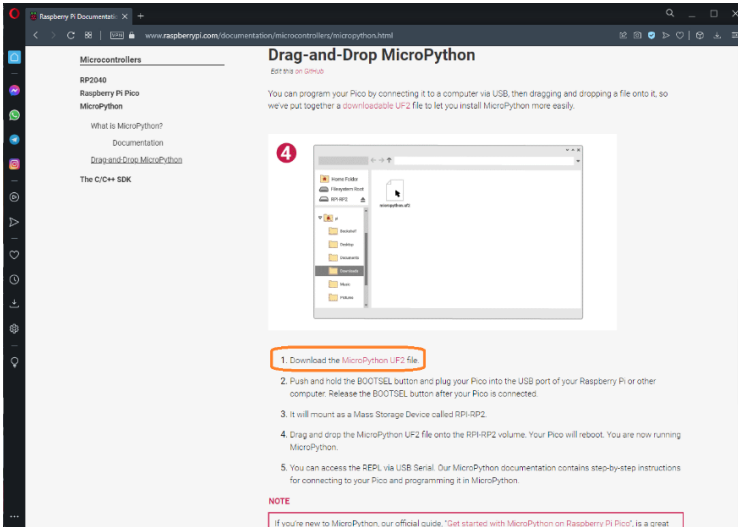
# Installing MicroPython Firmware on Raspberry Pi Pico

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.

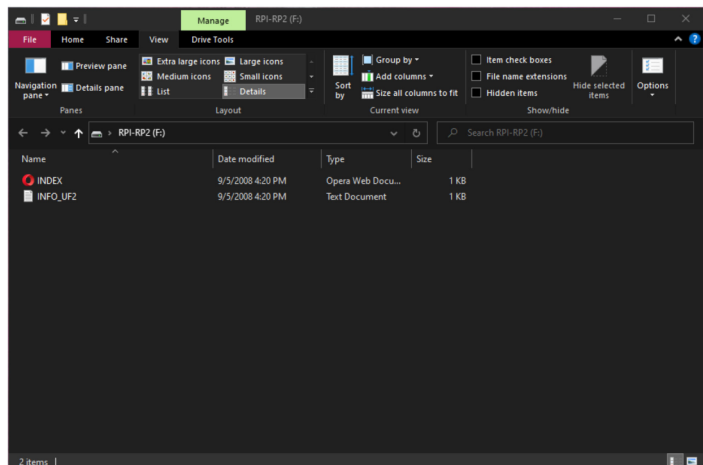
1- Just follow the instructions to download.

<https://www.raspberrypi.com/documentation/microcontrollers/micropython.html>

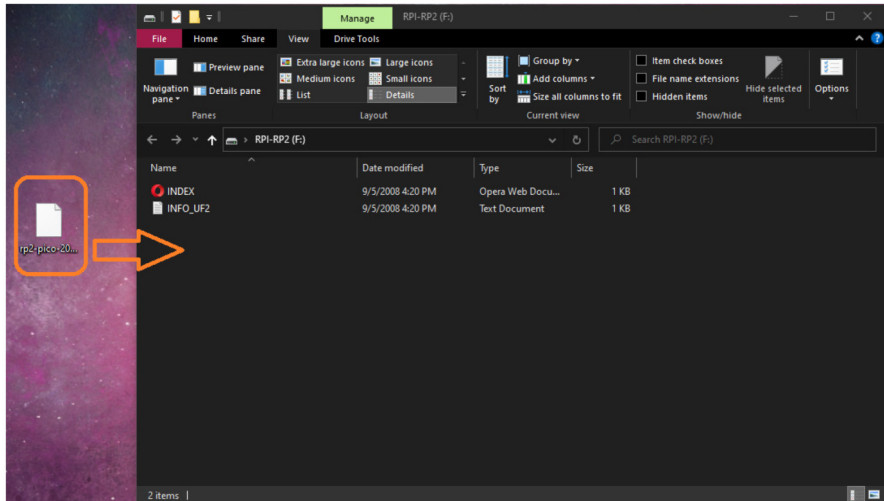
2- You click "Download UF2 file" and download the "firmware"



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

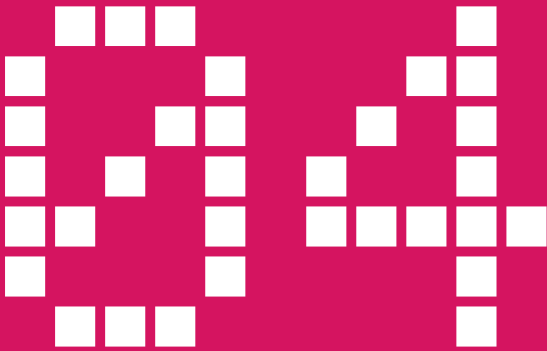


4- Copy the "\*.UF2" file that downloaded on that folder.



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

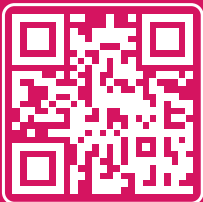




# Introducing Thonny IDE Interface

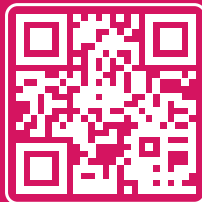


BLOG



[maker.robotistan.com](http://maker.robotistan.com)

FORUM



[forum.robotistan.com](http://forum.robotistan.com)



YouTube

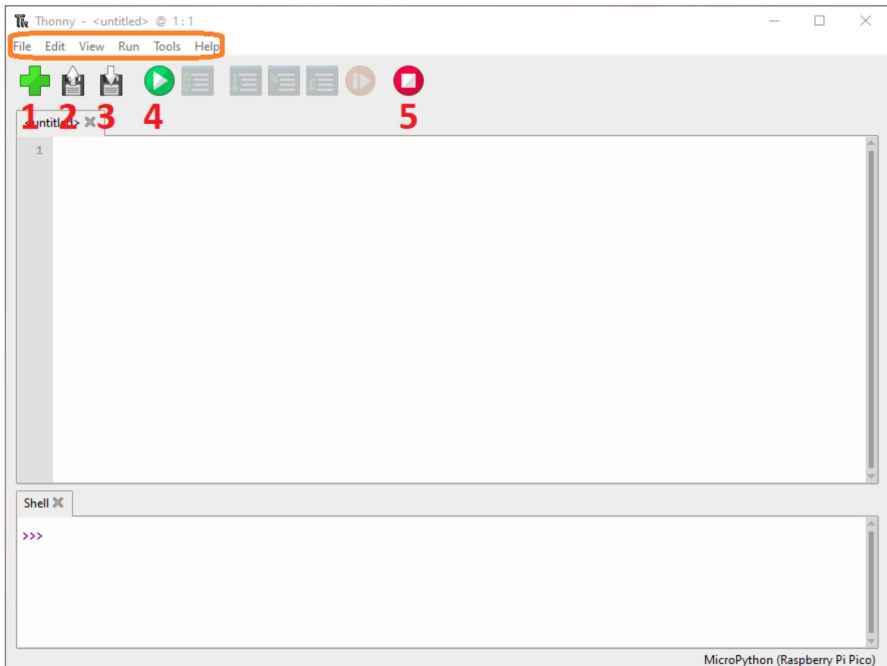


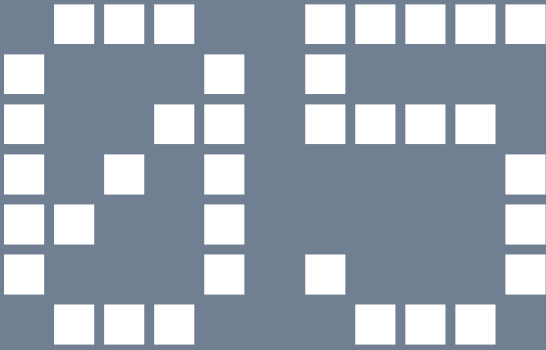
[youtube.com/robotistan](http://youtube.com/robotistan)

Unlike Arduino, Raspberry Pi Pico has its own memory, and library or scripts what you want can be uploaded into 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 Raspberry 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
5. You can stop the script with this icon





# Blink Internal LED



BLOG



[maker.robotistan.com](http://maker.robotistan.com)

FORUM



[forum.robotistan.com](http://forum.robotistan.com)



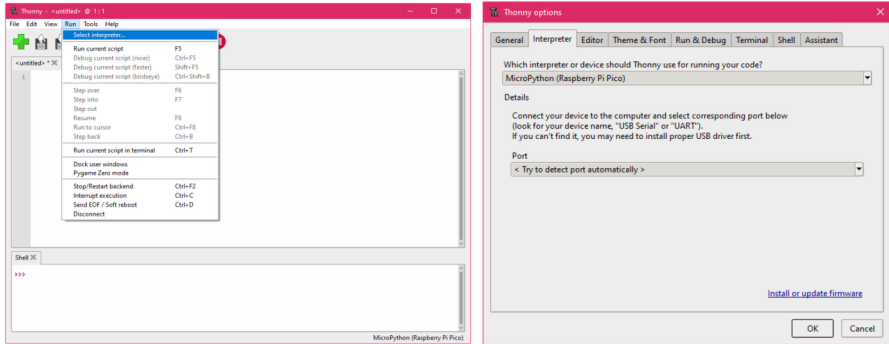
YouTube



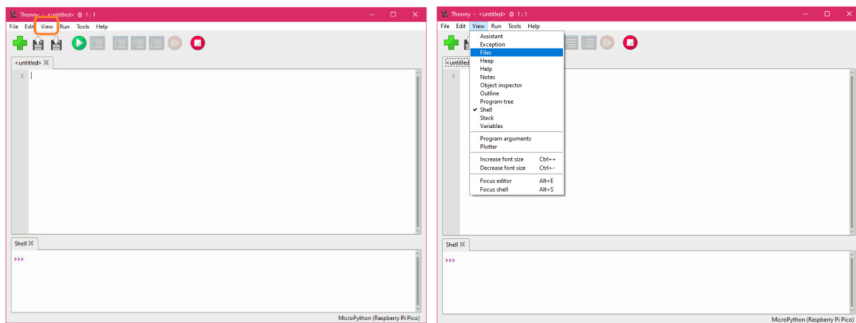
[youtube.com/robotistan](http://youtube.com/robotistan)

# Introducing Thonny IDE Interface

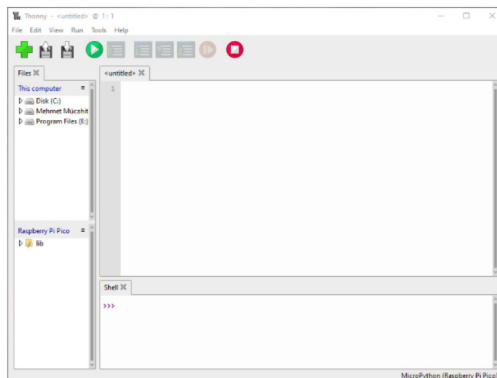
You adjust the Interpreter Settings on Thonny IDE for using your Raspberry Pi Pico. Click the the run tab at the top, then click Select Interpreter. Select "MicroPython (Raspberry Pi Pico)" as the "Interpreter" and "Try to detect port automatically" as the port from the window that opens.



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..





## Blink Internal LED

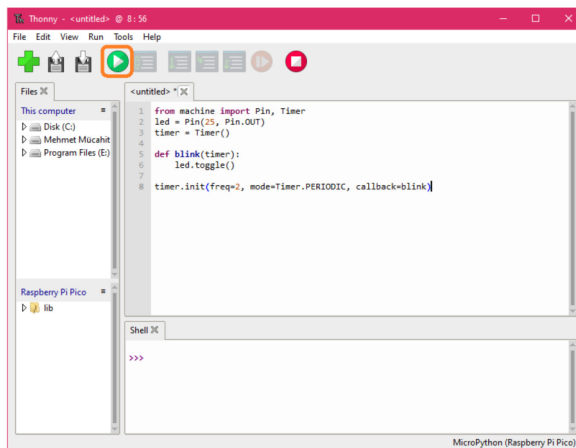
Type the following code into Thonny IDE this code provides the Internal LED to blink every half second When you examine this code, you wrote the code "from machine import Pin, Timer" so that be able to access the hardware on the Pico. In line 2, you stated that the internal LED on the Pico is connected 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()

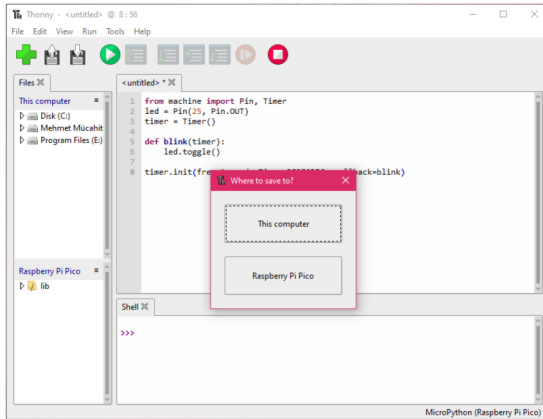
def blink(timer):
    led.toggle()

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

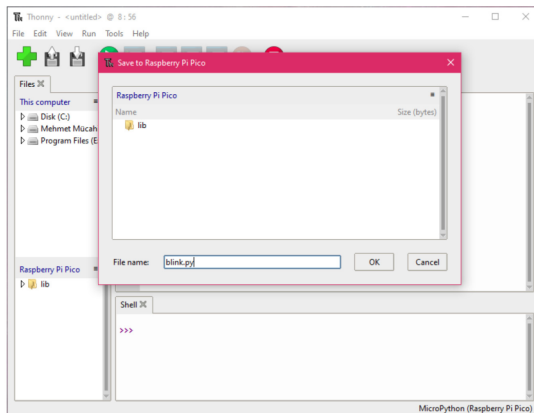
Upload this code to your Pico. Let's click on the green "Run" icon at the top.



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.

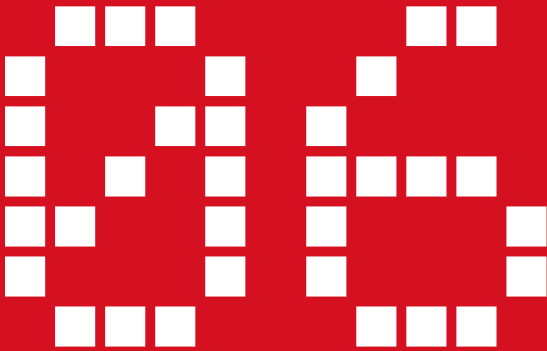


After clicking on the "Raspberry Pi Pico" text, a window like in the photo will open. In this window, it asks us to which location you want to save the code you wrote in Pico. You can save it directly in Pico. For this, you need to write a name and file extension to save the code you have written. Since 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.

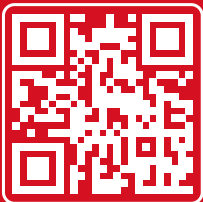




# LED Control with Button



BLOG



[maker.robotistan.com](http://maker.robotistan.com)

FORUM



[forum.robotistan.com](http://forum.robotistan.com)



YouTube



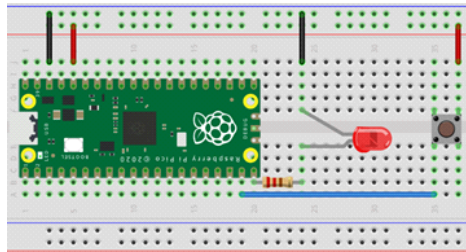
[youtube.com/robotistan](http://youtube.com/robotistan)

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  $\Omega$  or 330  $\Omega$  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.

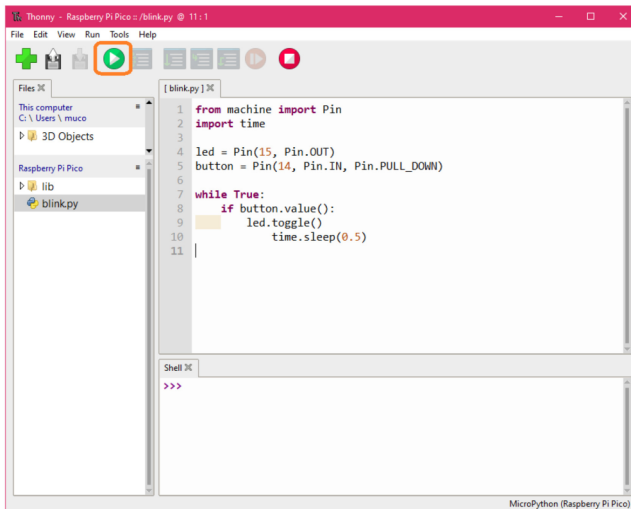
```
1 from machine import Pin, Timer
2 led = Pin(25, Pin.OUT)
3 timer = Timer()
4
5 def blink(timer):
6     led.toggle()
7
8 timer.init(freq=2, mode=Timer.PERIODIC, callback=blink)
9
10
```

## LED Control with Button

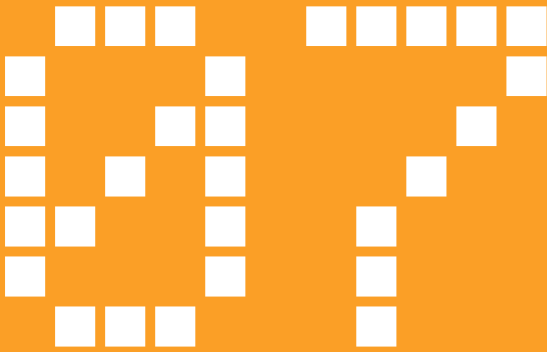
When you examine the codes below, you defined the hardware of Pi-co with the code "from machine import Pin" as in the previous project and added the "time" library. With the code "led = Pin(15, Pin.OUT)", you assigned the 15 nu-mara pin as the pin you connected to the LED. With the code "but-ton = Pin(14, 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.

```
1 from machine import Pin
2 import time
3
4 led = Pin(15, Pin.OUT)
5 button = Pin(14, Pin.IN, Pin.PULL_DOWN)
6
7 while True:
8     if button.value():
9         led.toggle()
10        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.



# Temperature Measurement with Pico



BLOG



[maker.robotistan.com](http://maker.robotistan.com)

FORUM



[forum.robotistan.com](http://forum.robotistan.com)



YouTube



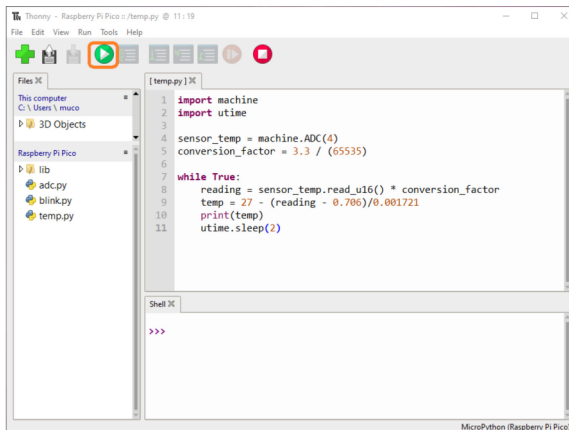
[youtube.com/robotistan](http://youtube.com/robotistan)

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 sensor 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 voltage. You convert the data you have obtained to "Celsius" in the 9th line. Finally, print the temperature data to Shell with the code "print(temp)".

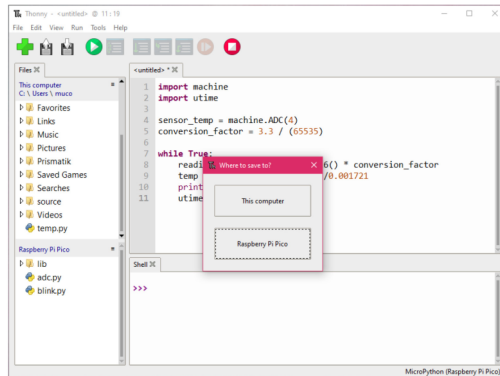
```
1 import machine
2 import utime
3
4 sensor_temp = machine.ADC(4)
5 conversion_factor = 3.3 / (65535)
6
7 while True:
8     reading = sensor_temp.read_u16() * conversion_factor
9     temp = 27 - (reading - 0.706)/0.001721
10    print(temp)
11    utime.sleep(2)
```

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

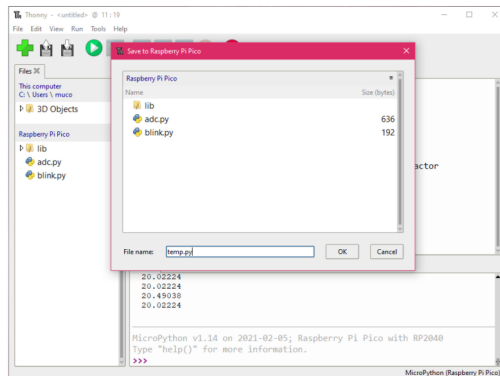


It asks where you want to save the code. For the code to always run, you need to click on the "Raspberry Pi Pico" option and save it to our Pico.

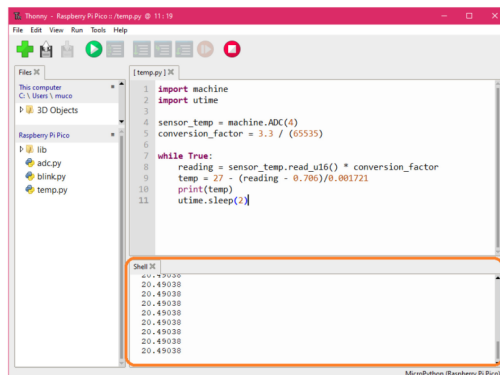
# Temperature Measurement with Pico



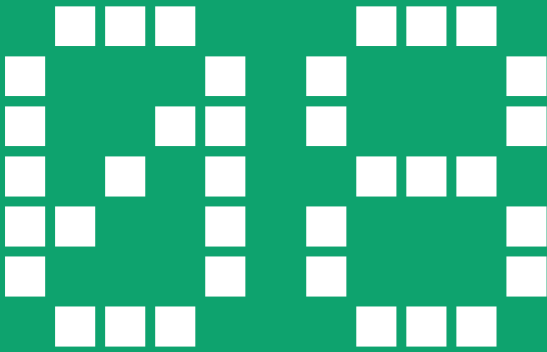
Write "temp.py" as the file name, click the "OK" button and now you can read the temperature data from Shell.



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



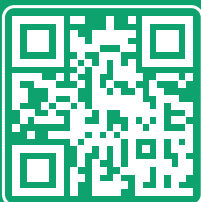




# Weather Monitor



BLOG



[maker.robotistan.com](http://maker.robotistan.com)

FORUM



[forum.robotistan.com](http://forum.robotistan.com)



YouTube



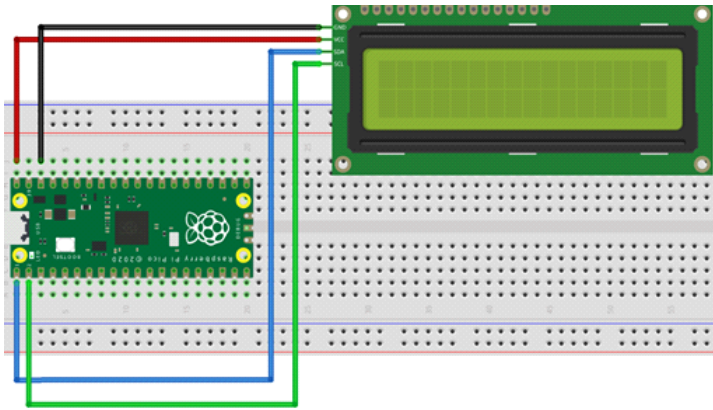
[youtube.com/robotistan](http://youtube.com/robotistan)

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.

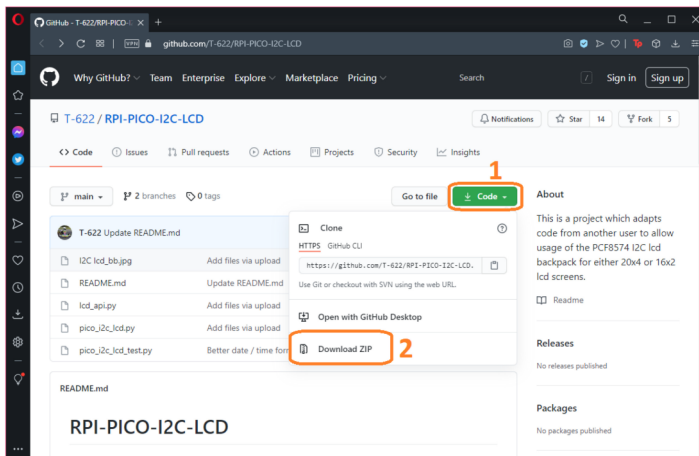
## Necessary materials:

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

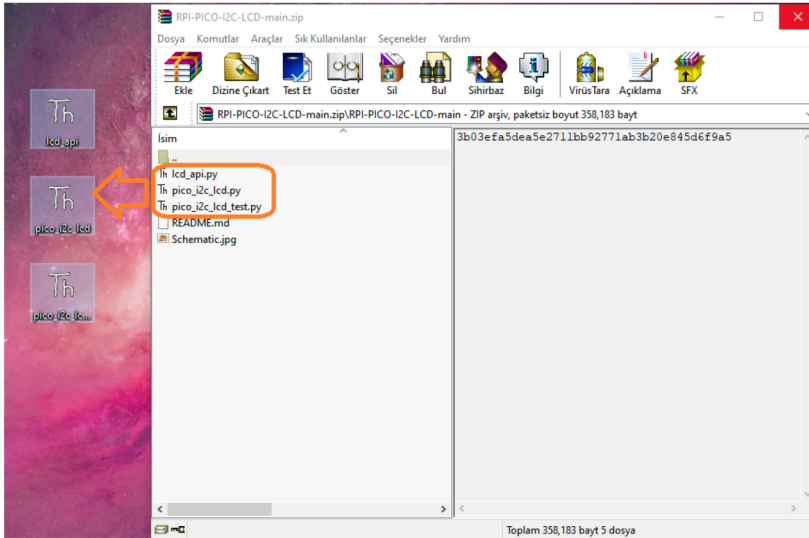
First, build your own circuit on the breadboard according to the circuit below.



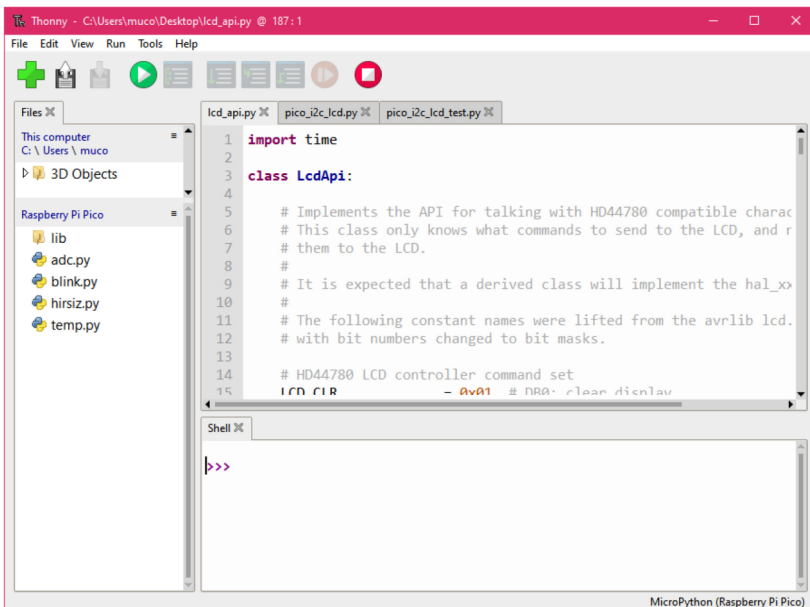
Now download the necessary library to use our LCD screen from <https://github.com/T-622/RPI-PICO-I2C-LCD>. When open the link, click the green button that says "Code" and then click the "Download ZIP" button to download the library.



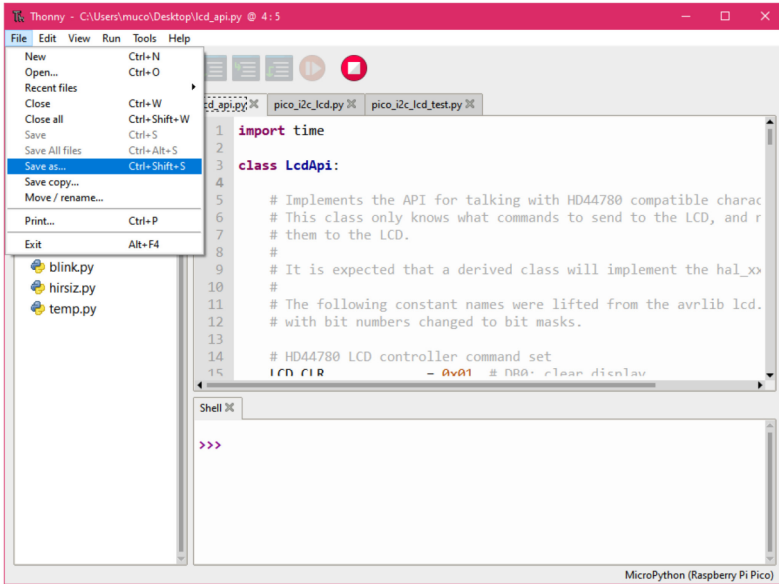
Open the downloaded "\*" .zip" file and extract the "lcd\_api.py", "pico\_i2c\_lcd.py", "pico\_i2c\_lcd\_test.py" files to the desktop.



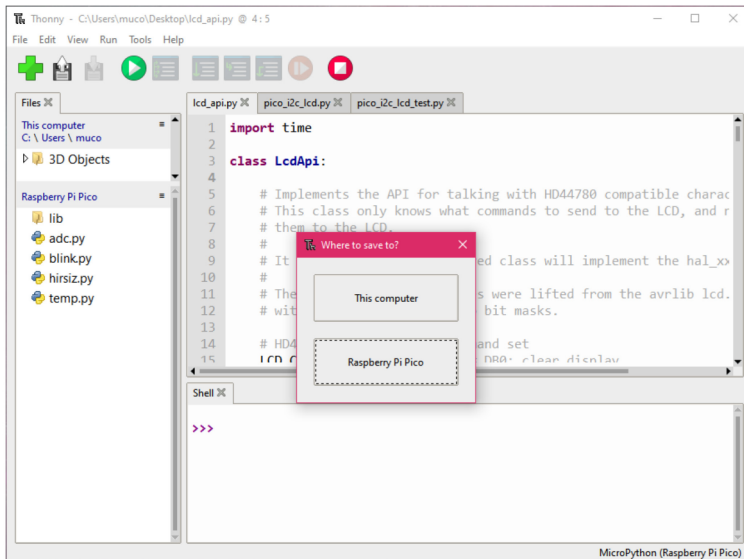
Open them in Thonny IDE by double clicking on these files that you extract to the desktop.



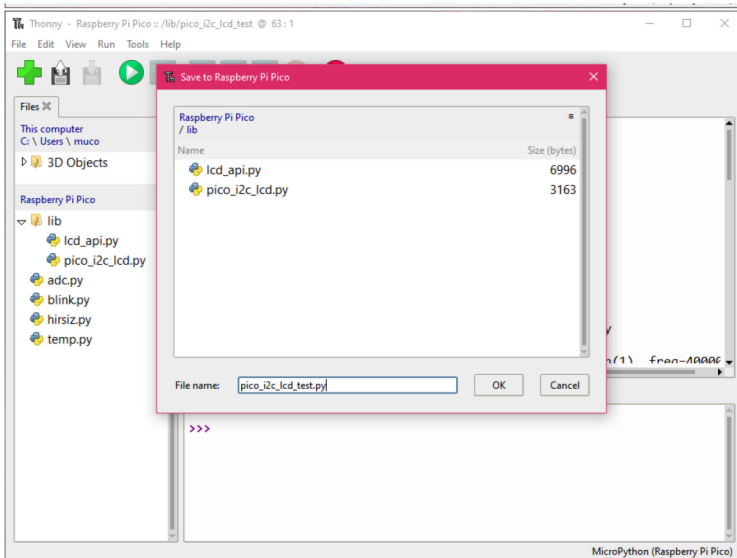
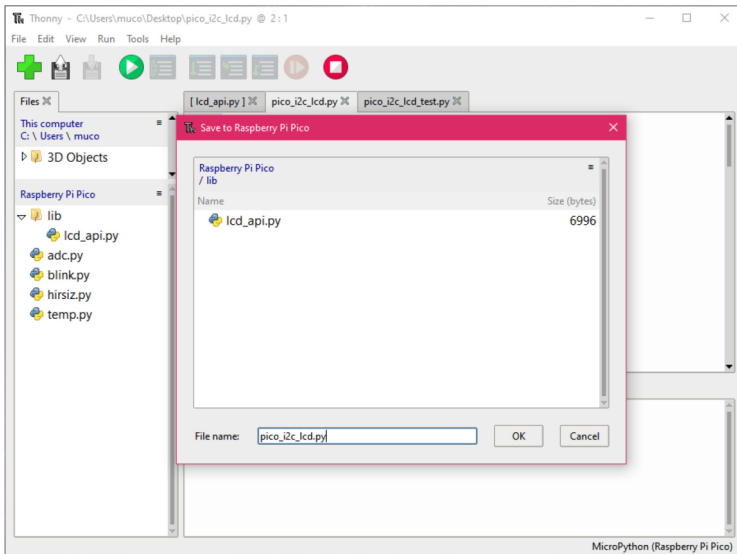
Now save these files to the "lib" folder in your Pico in order, click on the "Files" tab on the top and then click on "Save as ...".



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

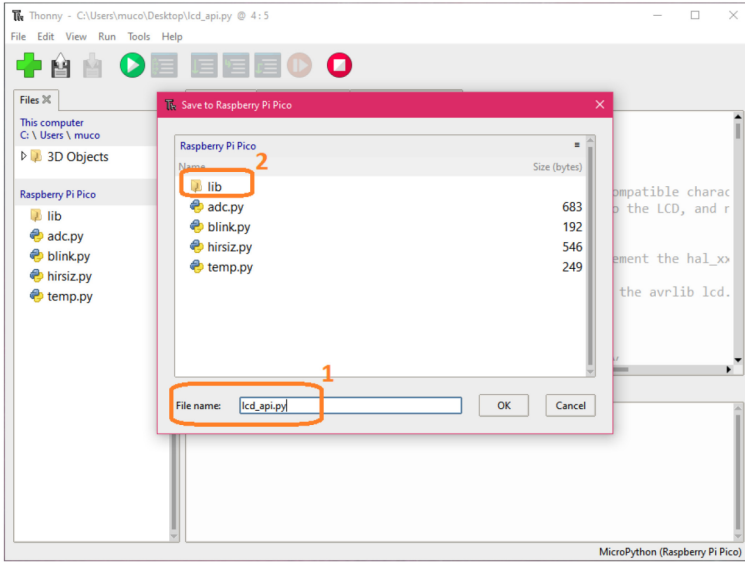


Do the same for the other two files.

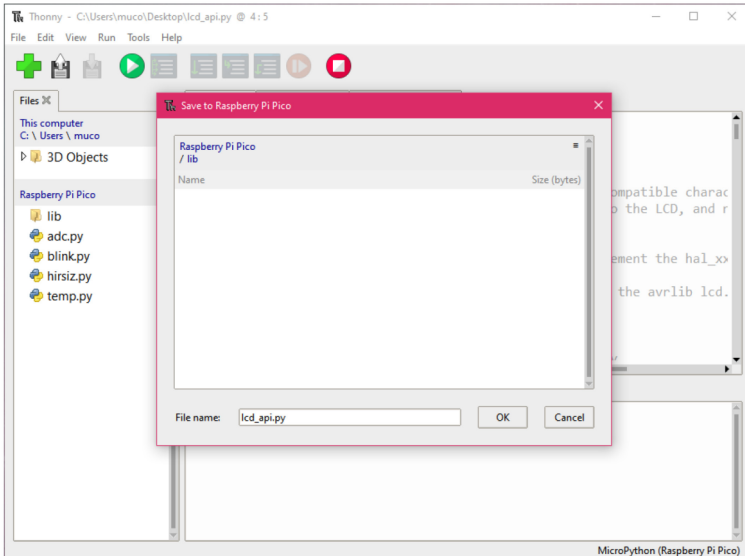


Now that you have added libraries, you can open a new file by clicking the "Plus" icon on the top of a new Thonny IDE and write the following codes. When you examine the codes, first added the libra-ries as in other projects, then made the settings of our 2X16 LCD screen and created a new text docu-ment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

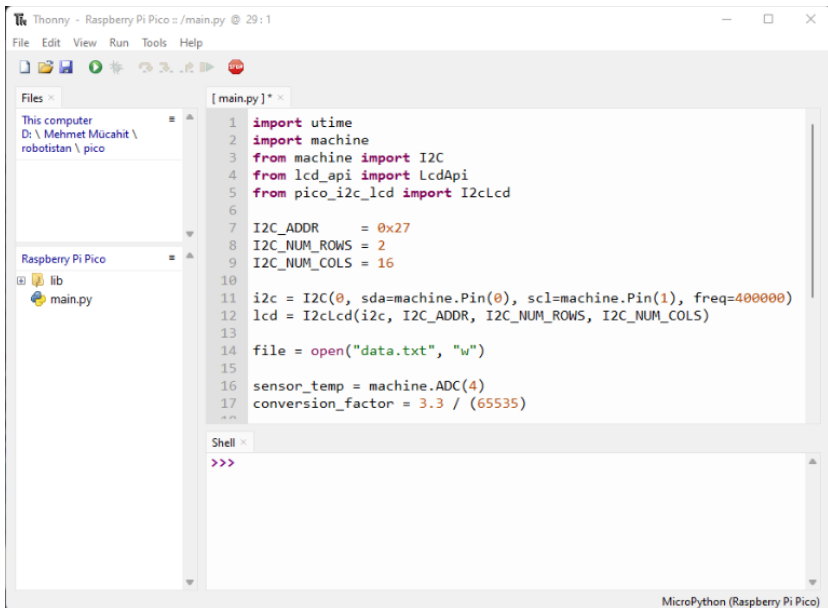


Then click the "OK" button to save it to the folder.

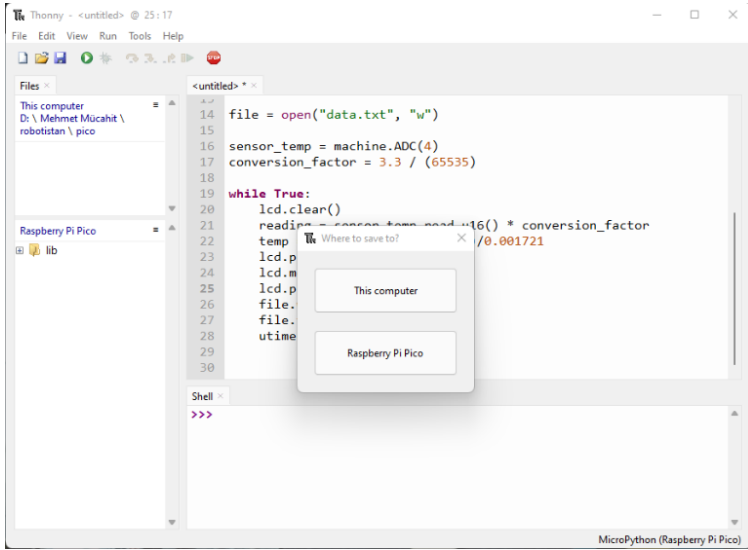


```
1 import utime
2 import machine
3 from machine import I2C
4 from lcd_api import LcdApi
5 from pico_i2c_lcd import I2cLcd
6
7 I2C_ADDR = 0x27
8 I2C_NUM_ROWS = 2
9 I2C_NUM_COLS = 16
10
11 i2c = I2C(0, sda=machine.Pin(0), scl=machine.Pin(1), freq=400000)
12 lcd = I2cLcd(i2c, I2C_ADDR, I2C_NUM_ROWS, I2C_NUM_COLS)
13
14 dosya = open("data.txt", "w")
15
16 sensor_temp = machine.ADC(4)
17 conversion_factor = 3.3 / (65535)
18
19 while True:
20     lcd.clear()
21     reading = sensor_temp.read_u16() * conversion_factor
22     temp = 27 - (reading - 0.706)/0.001721
23     lcd.putstr("Temp: ")
24     lcd.move_to(0,1)
25     lcd.putstr(str(temp))
26     dosya.write(str(temp) + "\n")
27     dosya.flush()
28     utime.sleep(2)
```

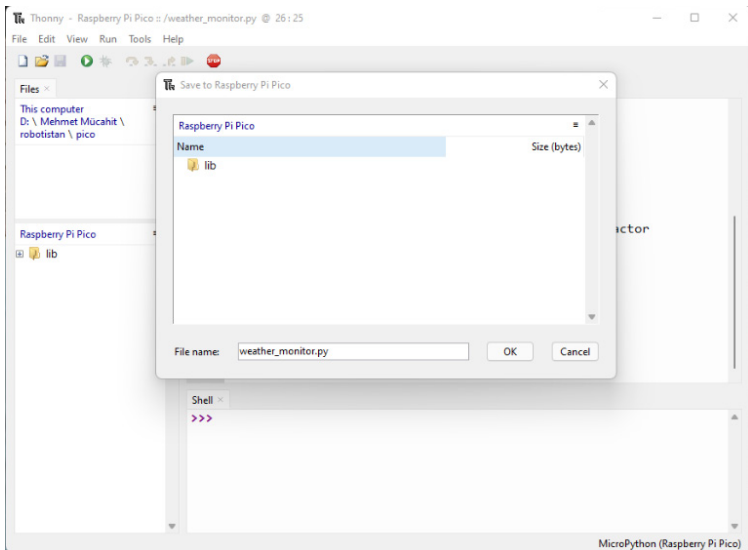
Now save these codes to your Pico and run them, then press the "Run" icon at the top.



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.

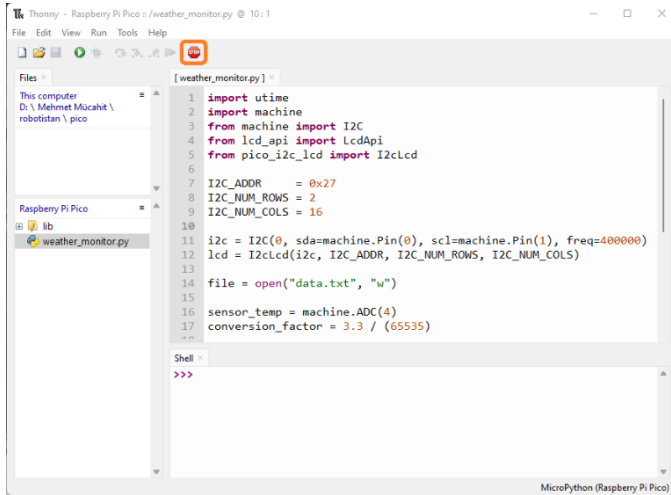


Type "weather\_monitor.py" as the file name and click the "OK" button. Now, you will be able to see the temperature data on your LCD screen and at the same time this data will be recorded in a text document.





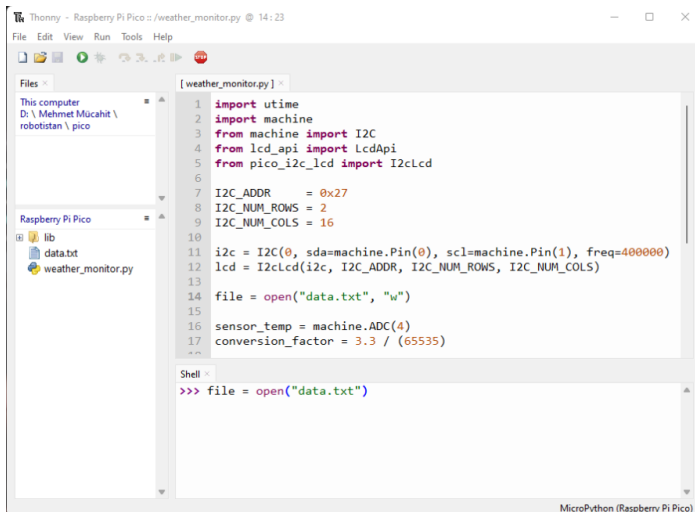
Now wait for some data to collect, then click the "Stop" icon at the top to stop the code from running.



```
1 import utime
2 import machine
3 from machine import I2C
4 from lcd_api import LcdApi
5 from pico_i2c_lcd import I2cLcd
6
7 I2C_ADDR = 0x27
8 I2C_NUM_ROWS = 2
9 I2C_NUM_COLS = 16
10
11 i2c = I2C(0, sda=machine.Pin(0), scl=machine.Pin(1), freq=400000)
12 lcd = I2cLcd(i2c, I2C_ADDR, I2C_NUM_ROWS, I2C_NUM_COLS)
13
14 file = open("data.txt", "w")
15
16 sensor_temp = machine.ADC(4)
17 conversion_factor = 3.3 / (65535)
```

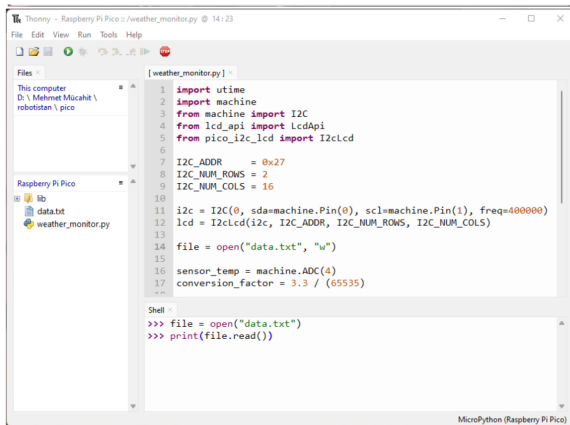
Write the following codes in the Shell section at the bottom so that you can see the data have saved.

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



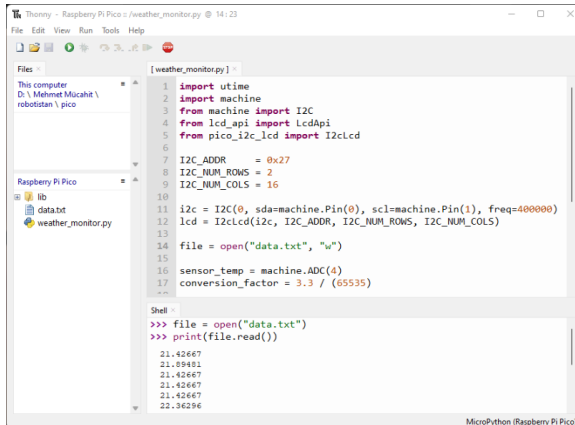
```
1 import utime
2 import machine
3 from machine import I2C
4 from lcd_api import LcdApi
5 from pico_i2c_lcd import I2cLcd
6
7 I2C_ADDR = 0x27
8 I2C_NUM_ROWS = 2
9 I2C_NUM_COLS = 16
10
11 i2c = I2C(0, sda=machine.Pin(0), scl=machine.Pin(1), freq=400000)
12 lcd = I2cLcd(i2c, I2C_ADDR, I2C_NUM_ROWS, I2C_NUM_COLS)
13
14 file = open("data.txt", "w")
15
16 sensor_temp = machine.ADC(4)
17 conversion_factor = 3.3 / (65535)
```

```
>>> file = open("data.txt")
```

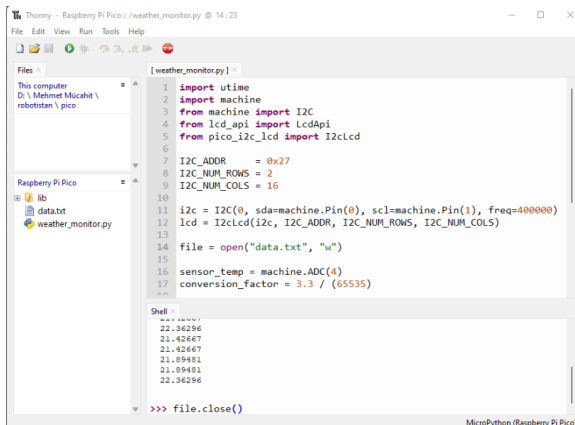


```
1 import utime
2 import machine
3 from machine import I2C
4 from lcd_api import LcdApi
5 from pico_i2c_lcd import I2cLcd
6
7 I2C_ADDR = 0x27
8 I2C_NUM_ROWS = 2
9 I2C_NUM_COLS = 16
10
11 i2c = I2C(0, sdamachine.Pin(0), scl=machine.Pin(1), freq=400000)
12 lcd = I2cLcd(i2c, I2C_ADDR, I2C_NUM_ROWS, I2C_NUM_COLS)
13
14 file = open("data.txt", "w")
15
16 sensor_temp = machine.ADC(4)
17 conversion_factor = 3.3 / (65535)
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

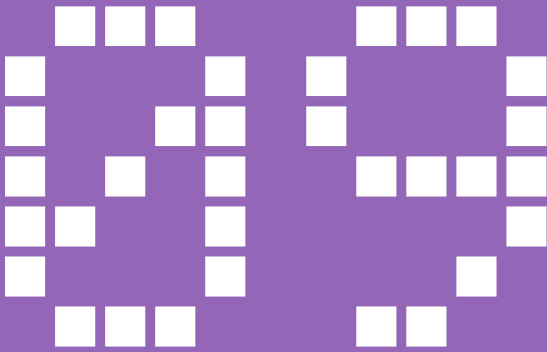
After typing the "print (file.read ())" code, you can see the saved data as in the photo below.



```
>>> file = open("data.txt")
>>> print(file.read())
21.42667
21.09481
21.42667
21.42667
21.42667
21.42667
22.36296
```



```
>>> file.close()
21.42667
21.42667
21.42667
21.42667
21.42667
21.42667
22.36296
```



# Burglar Alarm



BLOG



[maker.robotistan.com](http://maker.robotistan.com)

FORUM



[forum.robotistan.com](http://forum.robotistan.com)



YouTube



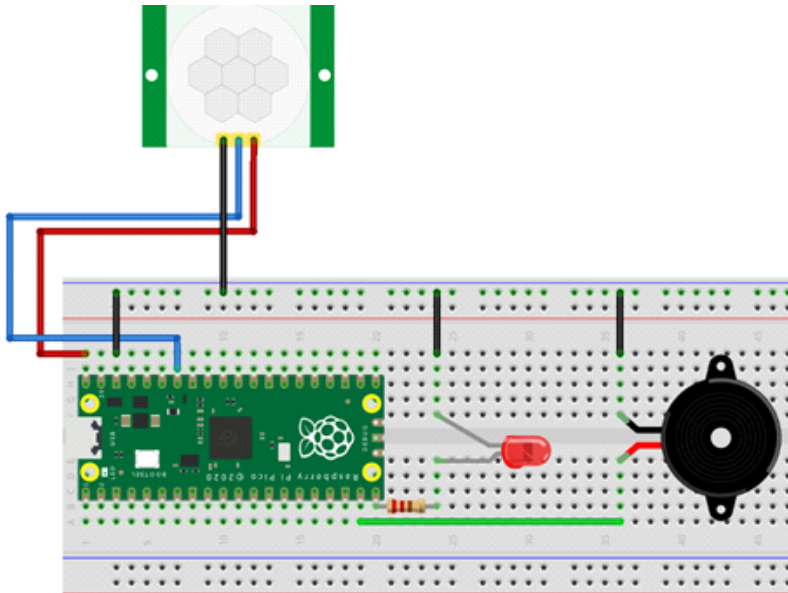
[youtube.com/robotistan](http://youtube.com/robotistan)

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  $\Omega$  or 330  $\Omega$  Resistor

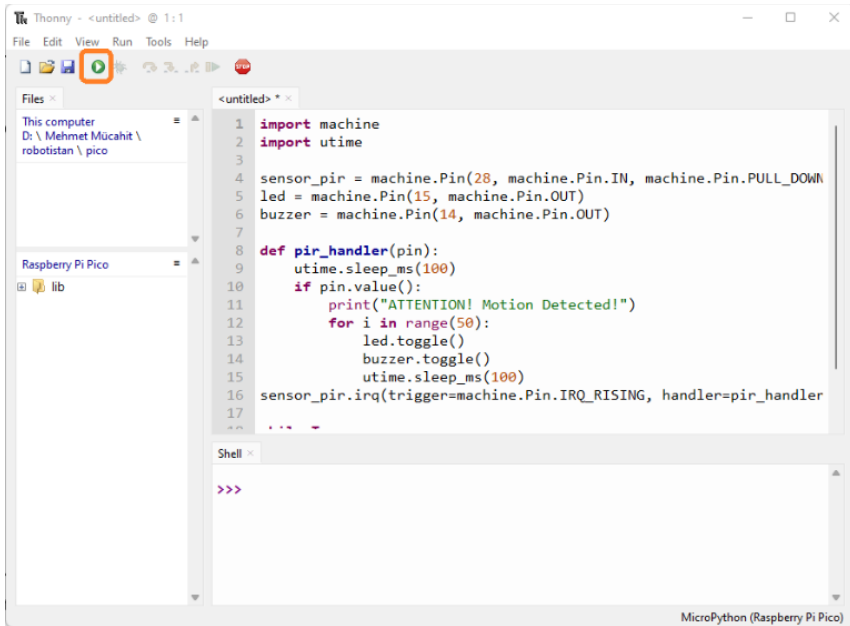
First, build your circuit according to the diagram below.



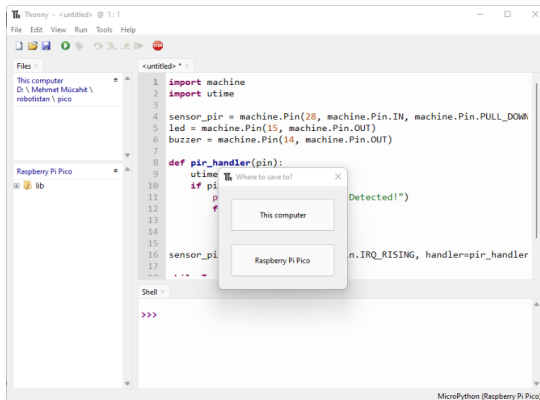
Now open a new file in Thonny IDE and write the following codes. . When you examine the codes, you first added the libraries and then defined the pins to which you connected the PIR sensor, Buzzer and LED. You created a function with "def pir\_handler (pin)". When the motion is detected with the if block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

```
1 import machine
2 import utime
3
4 sensor_pir = machine.Pin(28, machine.Pin.IN, machine.Pin.PULL_DOWN)
5 led = machine.Pin(15, machine.Pin.OUT)
6 buzzer = machine.Pin(14, machine.Pin.OUT)
7
8 def pir_handler(pin):
9     utime.sleep_ms(100)
10    if pin.value():
11        print("ATTENTION! Motion Detected!")
12        for i in range(50):
13            led.toggle()
14            buzzer.toggle()
15            utime.sleep_ms(100)
16 sensor_pir.irq(trigger=machine.Pin.IRQ_RISING, handler=pir_handler)
17
18 while True:
19     led.toggle()
20     utime.sleep(5)
```

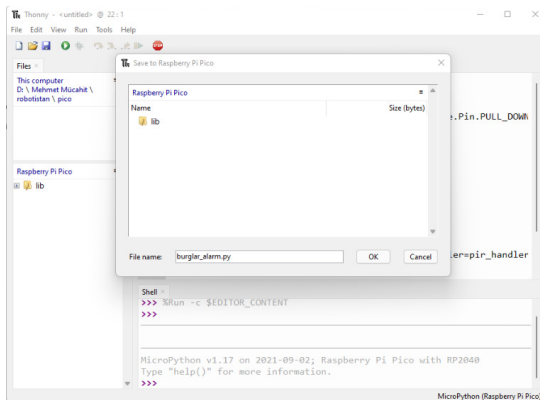
Now save these codes to your Pico and run them, then press the "Run" icon at the top.



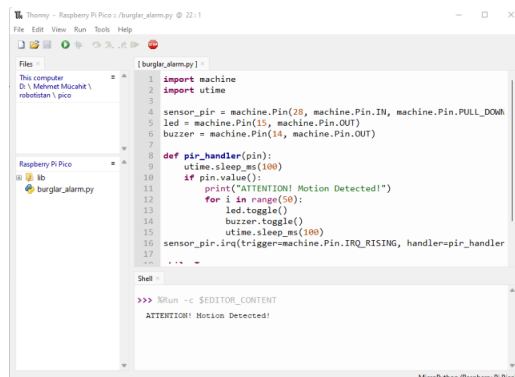
It asks where you want to save the code. For the code to always run, you need to click on the "Raspberry Pi Pico" option and save it to our Pico.

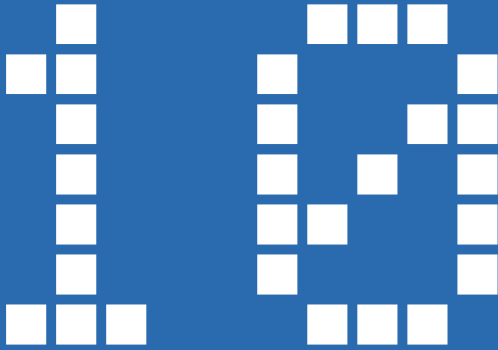


Write "burglar\_alarm.py" as the file name and click on the "OK" button and now, in case of any situation that triggers our motion sensor, the buzzer will sound and our LED will light.



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





# Using of Distance Sensor



BLOG



[maker.robotistan.com](http://maker.robotistan.com)

FORUM



[forum.robotistan.com](http://forum.robotistan.com)



YouTube



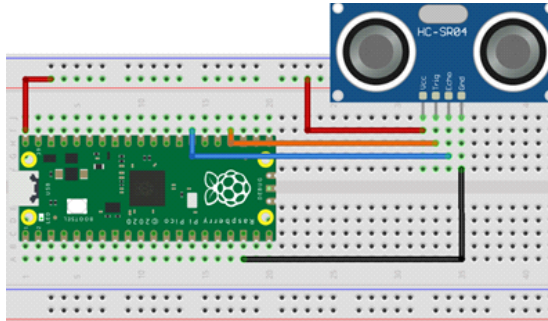
[youtube.com/robotistan](http://youtube.com/robotistan)

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.



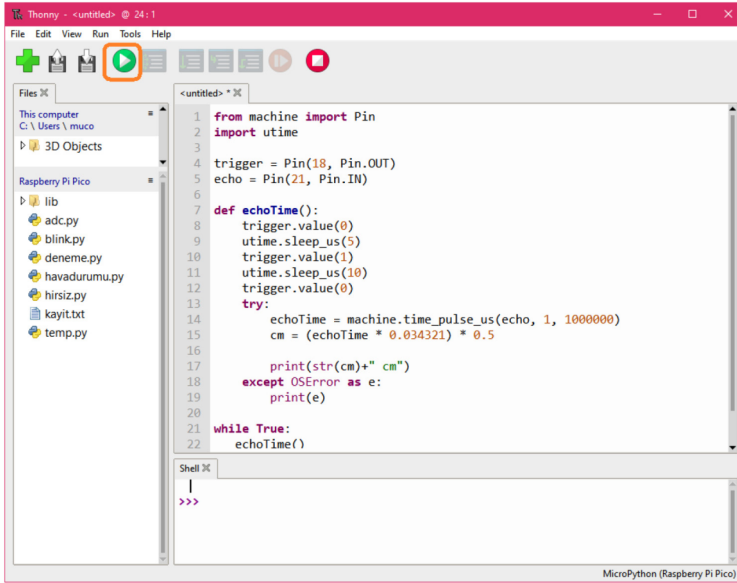
Now open a new file in Thonny IDE and write your code. When you examine the codes, you defined the libraries in the first two lines, and the pins to which the distance sensor is connected. Also wrote the distance measurement function with the function "def echoTime ():". You have made the distance measurement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

```
1 from machine import Pin
2 import utime
3
4 trigger = Pin(18, Pin.OUT)
5 echo = Pin(21, Pin.IN)
6
7 def echoTime():
8     trigger.value(0)
9     utime.sleep_us(5)
10    trigger.value(1)
11    utime.sleep_us(10)
12    trigger.value(0)
13    try:
14        echoTime = machine.time_pulse_us(echo, 1, 1000000)
15        cm = (echoTime * 0.034321) * 0.5
16
17        print(str(cm) + " cm")
18    except OSError as e:
19        print(e)
20
21 while True:
22     echoTime()
23     utime.sleep(5)
```

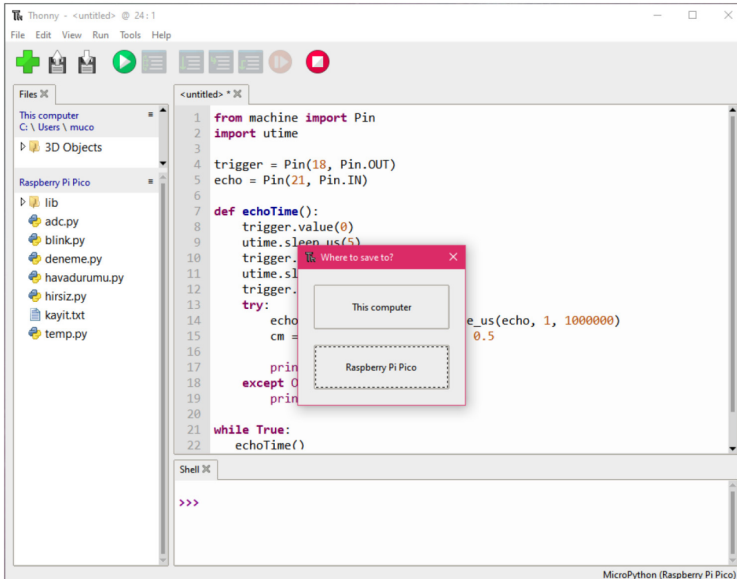


## Using of Distance Sensor

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

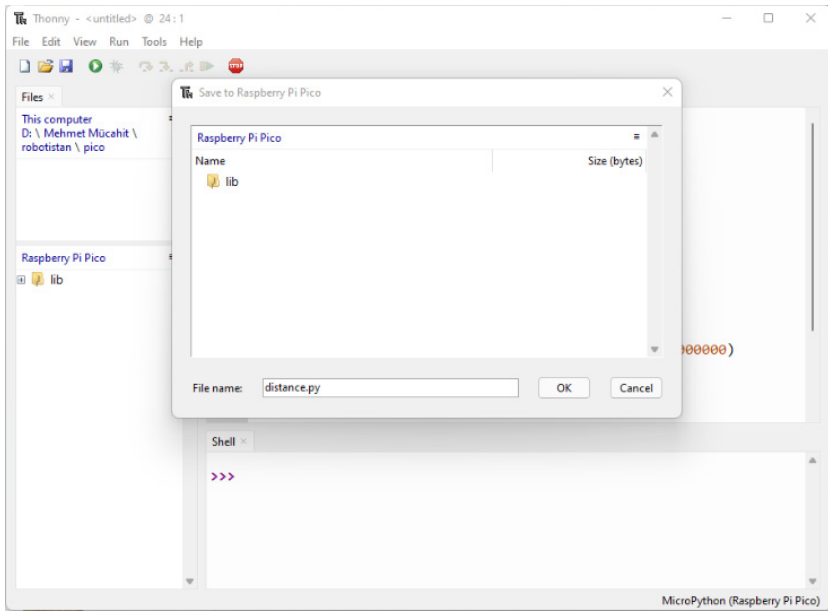


It asks where you want to save the code. For the code to always run, you need to click on the "Raspberry Pi Pico" option and save it to our Pico.

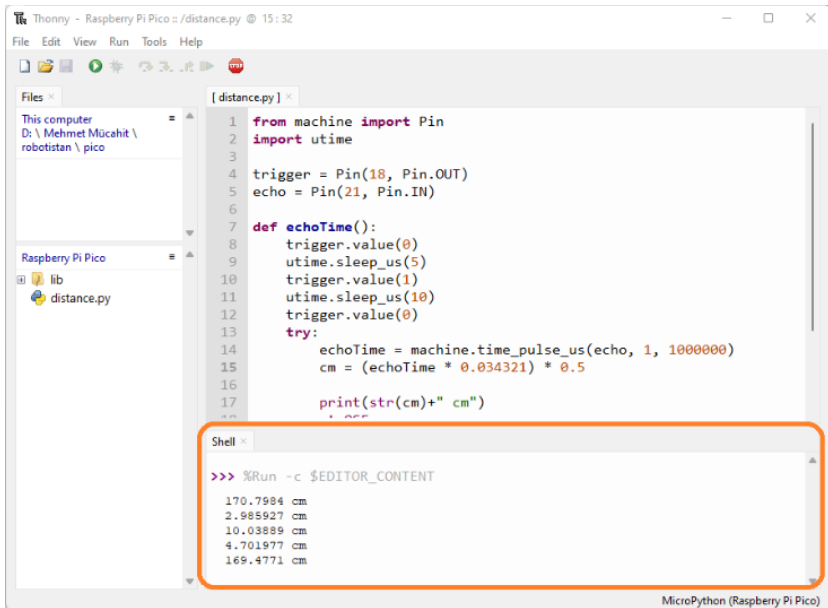


## Using of Distance Sensor

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



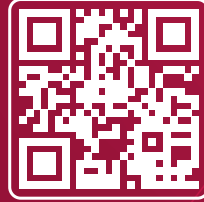
You can now see the distance in the Shell section of the Thonny IDE.





[youtube.com/robotistan](https://youtube.com/robotistan)

FORUM



[forum.robotistan.com](https://forum.robotistan.com)

BLOG



[maker.robotistan.com](https://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