

Data Logging With Raspberry Pi Pico

By Madhavan Thothadri (/member/Madhavan+Thothadri/) in Circuits (/circuits/) > Microcontrollers (/circuits/microcontrollers/projects/)

5,611

7

1



Download

Favorite

```
MicroPython v1.15 on 2021-04-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
```

```
>>> %Run -c $EDITOR_CONTENT
```

Data collection initiated.....

Collecting Sensor Data.....

Data fetched...

```
Logging data to the board....
```


Data logging complete!


>>>


[c:\Users\Alan.B.Fordman\Desktop\Bentley\Bentley_001\Bentley_001.dwg](#)

Logging data using a microcontroller sometimes becomes inevitable for projects and developments. Unlike the WIFI enabled NodeMCU and ESP32 microcontrollers which directly send data to a server, storing sensor or user data using an Arduino or any similar microcontroller is possible only with the use of an external SD card with its card reader.

In contrast, Raspberry Pi Pico microcontroller makes this job easy-peasy. Pi Pico enables smooth storing of sensor data on-board. As the Pico board works on flavors of Python (in my case Micro-python), a CSV file can be created precisely. Then, the data that is generated or sensed using any sensor can be appended to this CSV file concurrently. Thus all data is stored and it can further be transferred to the PC as when needed for analysis.

 Add Tip

 Ask Question

 Comment


Download


Supplies


Raspberry Pi Pico Microcontroller (Loaded with Micro-python)

PC with Thonny IDE

Micro-USB cable

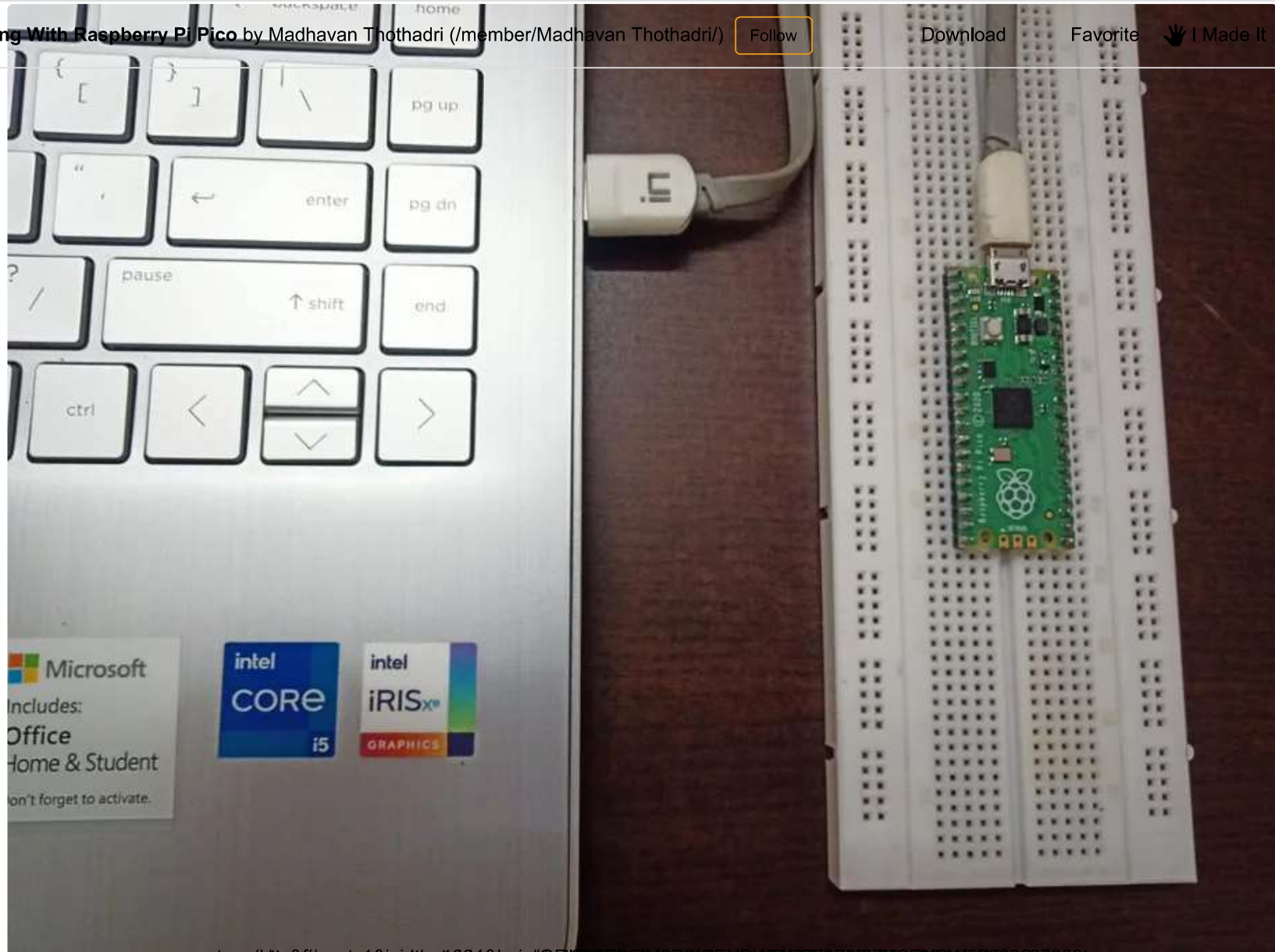
 Add Tip

 Ask Question

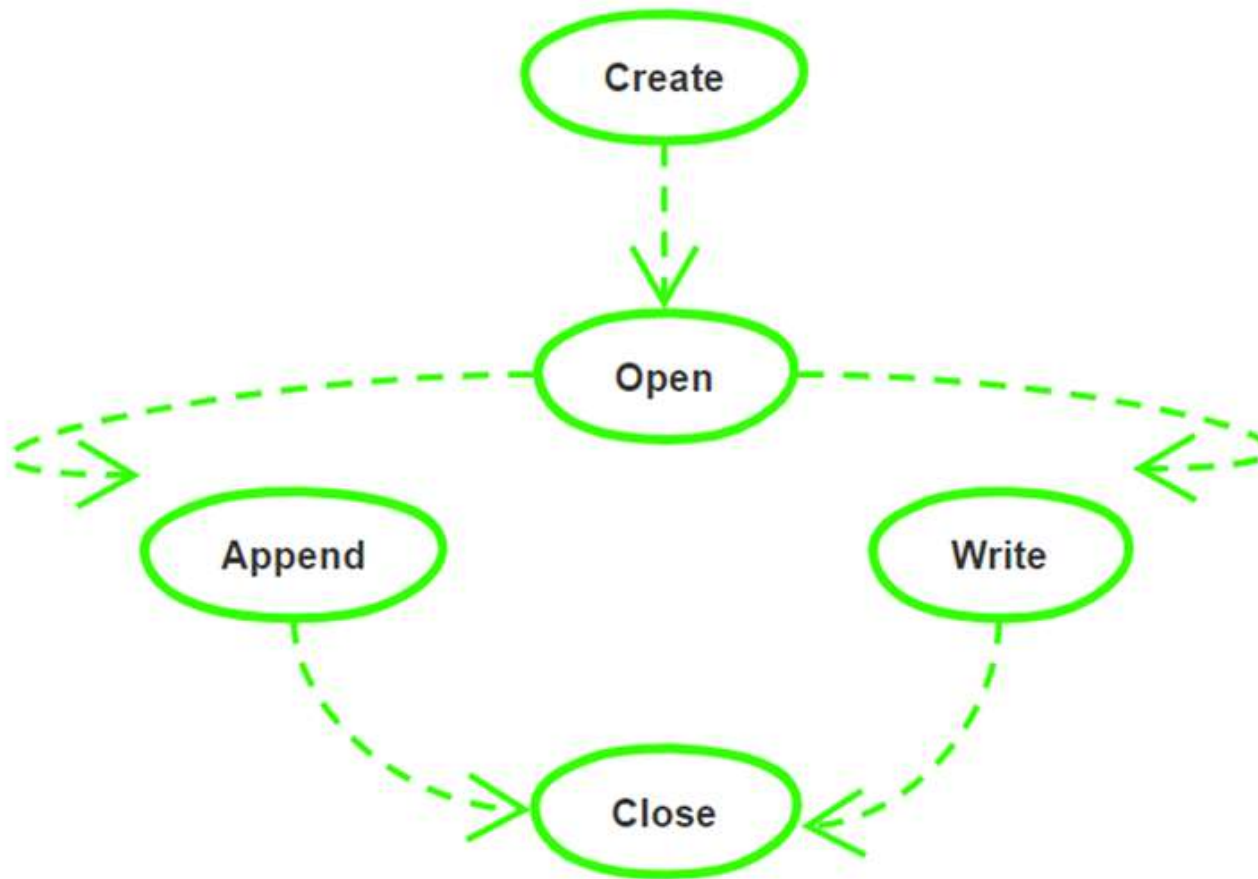
 Comment

Download

Step 1: The Setup



Connect the Pi Pico board to the PC using the Micro-USB cable . The board is Programmed with Thonny IDE. Once the IDE is opened and the port is enabled, the setup is complete.



<https://www.youtube.com/watch?v=1QnKtHh-10Q>

To store any data as a Comma Separated Values file in Python (and its flavors), the following hierarchy is followed.

- Creation of CSV file
- Opening of the file
- Writing data for the first time / Appending data
- Closing of CSV file.



Step 3: CSV File Creation, Opening and Saving

- In Micro-python, CSV file is created using the following syntax as

```
Objectname=open("filename.csv","mode")
```

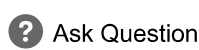
The function open() takes the filename along with its extension and its mode of opening as its parameters. In this case the file extension is **.csv**

If the specified file name already exists, the file gets opened. Otherwise, a new file gets created with the given filename and extension. The objectname can be any variable that is used for accessing the opened file.

The different modes of opening a file

- Write "w"
- Append "a"
- Read "r"
- Read-Write "r+"

Write mode is used for writing data onto a fresh file. In append mode, data gets added to the existing data. Using the Read mode, pre-existing data can be accessed but it cannot be modified. In Read-Write mode, the data can be read and tailored.



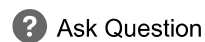
Step 4: Writing Data in the CSV File

Once the CSV file is created and opened in Write mode ("w"), the write() function is used for writing data.

Syntax

```
Objectname.write(str(value)+",")
```

Using the write function, the value to be stored is given as a string so as to add a separator " , " (Comma) at the end of each value. This process writes and stores the given data in CSV format. In the syntax, the function str() converts the given value into a string. The symbol "+" adds the separator comma (" , ") at the end of each string.

[Download](#)

Step 5: Closing of CSV

On adding all the values to the created CSV file, the file gets closed automatically as soon as the program is terminated. If needed, the file can be explicitly closed during execution with this syntax.

```
Objectname.close()
```

If the file is not closed explicitly, then the flush function needs to be called to clear the internal buffer. On the other hand, the internal buffer gets flushed automatically with the use of close() function .

Syntax for flush() function

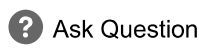
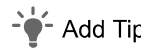
```
Objectname.flush()
```



Step 6: The Code

With all the syntax, the complete code for data logging is given here.

```
file=open("data.csv","w")      # creation and opening of a CSV file in Write mode
# Type Program Logic Here
file.write(str(value)+",")      # Writing data in the opened file
# file.flush()                 # Internal buffer is flushed (not necessary if close() function is used)
file.close()                   # The file is closed
```



Step 7: Example

Logging a simple math operated data to the Pico board

The following code shows a simple example of computing and storing the multiples of 5, with the starting value 95 by cumulative subtraction. The values are stored in a CSV file with the filename "add.csv".

```
a=95
file=open("add.csv","w")      # file is created and opened in write mode
while a>0:                    # program logic
    file.write(str(a)+",")    # data is written as a string in the CSV file
    file.flush()              # internal buffer is flushed
    a-=5
```