

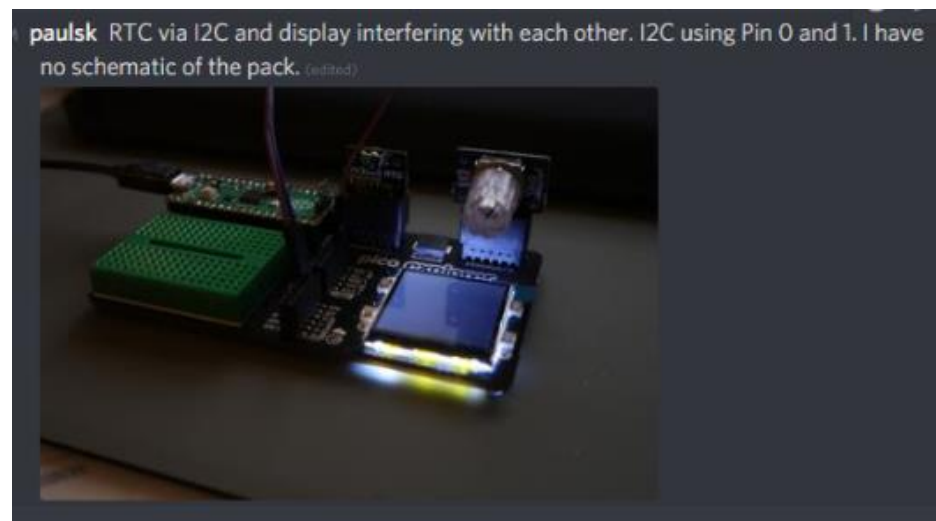
This is a report about my first C++ project for a Raspberry Pi Pico that I created from scratch and finished successfully. I make this report because two months after finishing the project; the app/clock is running impecable (even when not been powered on for days); I was asking myself which version I have running in the Raspberry Pi that is running the C++ application using an external realtime clock device of the RV3028 type. So, I found out. Now it's time to write this report and then publish this project on GitHub. I assume that the reader knows how to install and use the needed C++ toolchain for the Pico, has knowledge of C++ and knows how to use the various tools of the toolchain to be used, although I will describe the steps I do for starting the tools and issuing the build commands, in this project's case.

How it started

As far as I can see in some pre-project trial folders, I see that in March 2021 I was doing some trials with a Raspberry Pi Pico in MicroPython/CircuitPython using the Thonny IDE.

I found an issue I created on March 18, 2021, on GitHub concerning an error I encountered with the I2Cdevice library.

I also found the following image of a question I posted in Discord on March 20:



The problem I faced at that moment was because I started too quickly experimenting after I received the ordered a few Picos. After reading some documentation about the Pico I thought that GPIO0 and GPIO1, beside for UART0 were used for I2C, but I had not really read the text on the back of the Pimoroni pico explorer base where is printed a list of 'PICO PIN' and their 'FEATURE's. This list says: GP20 I2C SDA and GPIO21 I2C SCL. It was after reading a file of another repository on GitHub that 'opened my eyes' towards pins 20 and 21.

In that start period I studied the RV3028 and I2Cdevice library files. At a certain point I also added functions like: IsLeapYear() and DaysInMonth() and some other functionalities that I don't recall in this moment.

I made also modifications to the I2Cdevice-python.py and the rv3028-python.py file (that came from the Pimoroni GitHub repository (i2cdevice-pythonⁱ and rv3028-pythonⁱⁱ)).

I don't remember why I made the 'move' from building the project for the external rtc module from Python to C++. When I now think about it, it was because of memory problems. I remember that, in the CircuitPython environment I got into memory problems. The IDE reporting errors that appeared odd to me, not logical. It appeared that the program was loosing it's way in memory or overwriting it ending up in reporting errors of a type like 'variable not found' or 'not declared'. Something like that. Beside that fact it was the fact that I had seen a lot of examples (in fact the whole pico-sdk) written in C++ that I thought: I want to build a project in C++. So I entered the C++ 'road'. It was a big 'adventure', at some point I descriped my journey into C++ as a 'battle from line to line'. At several moments I became very frustrated about banging my head constantly to the C++ 'wall'. When I wanted to go 'left' C++ told me: 'no, you can't do that: 'you're trying to make apples from pears'. Then, with this 'knowledge' I turned 'right', but again C++ was patronizing me and telling me: 'no, Paulus, you can't do that for ... (reason)'. At a moment I was so desperate that I bought a kindle e-book about C++ and studied for a while. Coming with quite a backpack of Python experience, within C++ I felt as being in a small cage. I felt that, as a programmer/developer, I had much less freedom within C++. This was a 'feeling' which in fact isn't true. There are so many applications out in the world, all written in C++. No, sure, it was my lack in knowledge of C++. I never learned C++ from the basics. Anyway. I was proceeding on my C++ endeavouring voyage and I thought: you have to continue your struggle, move on! Don't give up! Beside C++ there was another big 'unknown' to me and that was the language of 'CMake' which

is used a lot for building C++ projects. CMake was another 'headbanger' for me. So, I also took a short online course in how to use CMake. This helped. I began to be able to compile and build C++ programs. I used Microsoft's VSCode IDE for development, testing and building. I became very fond of VSCode. For me it is the best development tool I used until now. I love the 'Outline' function, that builds a list of your functions, object, variables and so on. I also like very much that 'minimap' that one can activate from the 'View' menus and which 'minimap' shows on the right side via which one can scroll quickly through the codelines of the file one is editing.

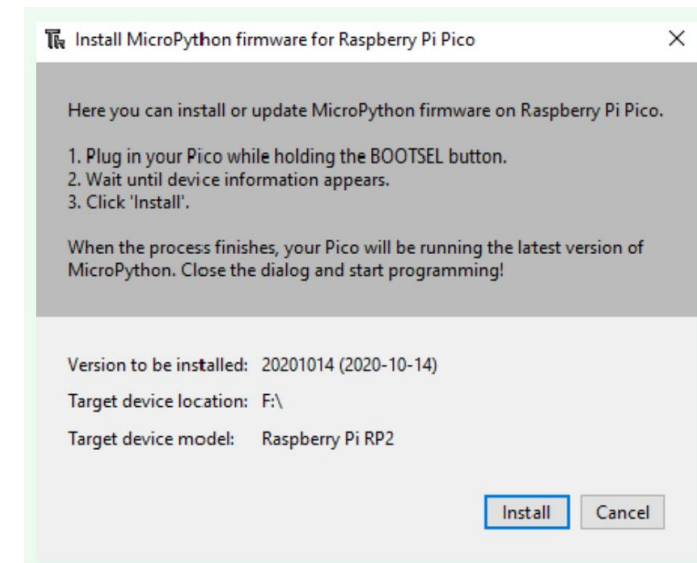
So far the 'story' on my adventure into this project. Now the real part.

The hardware

- One Raspberry Pi Pico ⁱⁱⁱ;
- a Pimoroni pico explorer base ^{iv};
- a Pimoroni RV3028 breakout ^v;
- a USB-A to microUSB cable.

The software

- Raspberry Pi org firmware for the Raspberry Pi Pico (can be installed from within the Thonny IDE. See the screenshot to the right)^{vi};
- Raspberry Pi org pico-sdk^{vii};
- RV3028 library from Constantin Koch on GitHub^{viii};
- Pimoroni C++ libraries, part of GitHub repository^{ix}:
 - o Pico explorer;
 - o Pico graphics;
 - o st7789 LCD;
- Microsoft Visual Studio Code^x;
- Microsoft Windows Subsystem for Linux, version 1 (because that makes connecting COM ports available)^{xi};
- Ubuntu v 20.04 for WSL1 (installed from the Microsoft Store);
- Various extensions to VSCode: (C/C++, CMake, CMake-Tools);



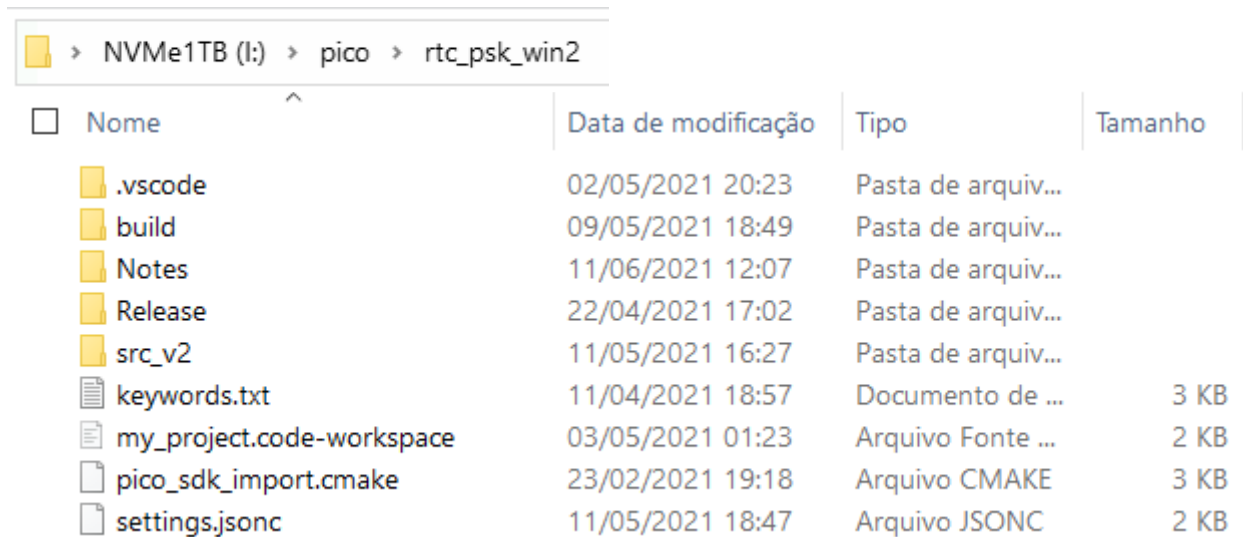
Setting up the toolchain

I followed all the steps to setup the toolchain to develop C++ projects for the RPi Pico using the pico-sdk as described in the documentation that the Raspberry Pi organization has published. When one puts the RPi Pico in the 'Bootsel' mode, there will popup a window showing a disk with the name "RPI-RP2". This disk shows two files. One is named: "INDEX.HTM". When one clicks on this file a webbrowser app session is started and redirected to the webpage "RP2040". On that webpage are a list of links that link to various documents. The one to start with is: 'Getting started with Raspberry Pi Pico'^{xii}. Subtitle: "C/C++ development with Raspberry Pi Pico and other RP2040-based microcontoller boards". This document learns how to get the SDK and examples; how to install and update the toolchain; build, flash and run the first examples.

Another good source for one's Pico C++ projects is: 'Raspberry Pi Pico C/C++ SDK'. Clicking on this link will open a document with the same name and subtitle: "Libraries and tools for C/C++ development on RP2040 microcontrollers".^{xiii}

Project file structure

The following section is maybe a bit 'boring', just a list of screenshots taken from various project subfolders and subfolders of used tools like pico-sdk and the pimoroni-pico repository. **Note** that on GitHub the name of the root folder is: '[rpi_pico_cpp_ext_rtc](#)'.


































<input type="checkbox"/> Nome	Data de modificação	Tipo	Tamanho
 .vscode	02/05/2021 20:23	Pasta de arquiv...	
 build	09/05/2021 18:49	Pasta de arquiv...	
 Notes	11/06/2021 12:07	Pasta de arquiv...	
 Release	22/04/2021 17:02	Pasta de arquiv...	
 src_v2	11/05/2021 16:27	Pasta de arquiv...	
 keywords.txt	11/04/2021 18:57	Documento de ...	3 KB
 my_project.code-workspace	03/05/2021 01:23	Arquivo Fonte ...	2 KB
 pico_sdk_import.cmake	23/02/2021 19:18	Arquivo CMAKE	3 KB
 settings.jsonc	11/05/2021 18:47	Arquivo JSONC	2 KB














Image to the left:

The projects root folder.

 > NVMe1TB (I:) > pico > rtc_psk_win2 > src_v2				
<input type="checkbox"/> Nome	Data de modificação	Tipo	Tamanho	
 app_v2	11/05/2021 10:08	Pasta de arquiv...		
 pico_explorer	05/05/2021 19:34	Pasta de arquiv...		
 pico_graphics	05/05/2021 19:34	Pasta de arquiv...		
 psk_hw	11/05/2021 17:16	Pasta de arquiv...		
 rv3028	05/05/2021 19:34	Pasta de arquiv...		
 st7789	05/05/2021 19:34	Pasta de arquiv...		
 CMakeLists.txt	06/05/2021 01:02	Documento de ...	1 KB	
 pico_sdk_import.cmake	23/02/2021 19:18	Arquivo CMAKE	3 KB	

 > NVMe1TB (I:) > pico > rtc_psk_win2 > src_v2 > app_v2				
<input type="checkbox"/> Nome	Data de modificação	Tipo	Tamanho	
 app_v2 - Copia.cpp	07/05/2021 00:34	Arquivo Fonte ...	39 KB	
 app_v2.1.cpp	11/05/2021 20:53	Arquivo Fonte ...	66 KB	
 app_v2.cpp	17/05/2021 15:14	Arquivo Fonte ...	70 KB	
 app_v2.hpp	17/05/2021 15:27	Arquivo Fonte ...	15 KB	
 app_v2_second.cpp	09/05/2021 10:41	Arquivo Fonte ...	54 KB	
 CMakeLists.txt	24/05/2021 21:54	Documento de ...	1 KB	

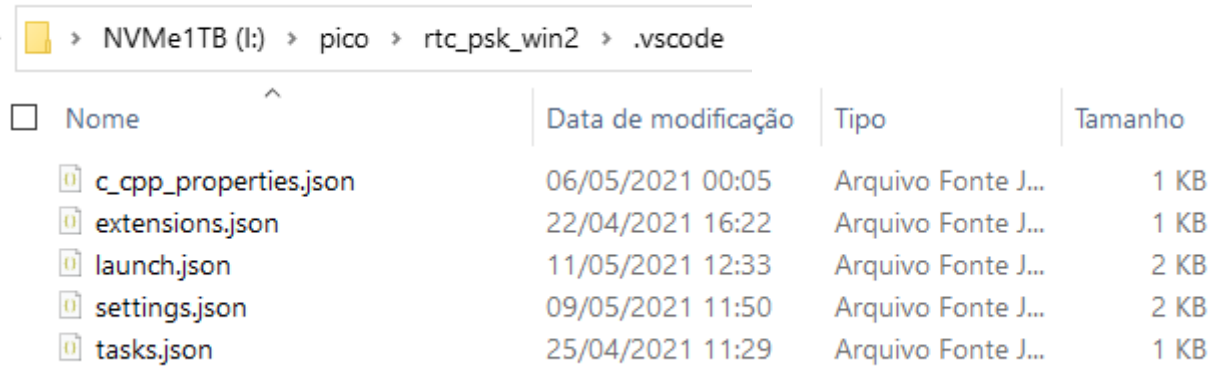
 > NVMe1TB (I:) > pico > rtc_psk_win2 > src_v2 > pico_explorer			
<input type="checkbox"/> Nome	Data de modificação	Tipo	Tamanho
 CMakeLists.txt	27/04/2021 15:58	Documento de ...	1 KB
 pico_explorer.cpp	11/05/2021 20:58	Arquivo Fonte ...	4 KB
 pico_explorer.h	26/04/2021 18:57	Arquivo Fonte ...	2 KB
 pico_explorer.hpp	26/04/2021 20:56	Arquivo Fonte ...	2 KB
 README.md	23/02/2021 19:18	MD Documento	6 KB






 > NVMe1TB (I:) > pico > rtc_psk_win2 > src_v2 > pico_graphics			
<input type="checkbox"/> Nome	Data de modificação	Tipo	Tamanho
 CMakeLists.txt	27/04/2021 15:59	Documento de ...	1 KB
 font.hpp	27/04/2021 17:46	Arquivo Fonte ...	1 KB
 font_data.hpp	23/02/2021 19:18	Arquivo Fonte ...	5 KB
 font6_data.hpp	26/04/2021 20:31	Arquivo Fonte ...	5 KB
 font8_data.hpp	26/04/2021 20:32	Arquivo Fonte ...	5 KB
 pico_graphics.cmake	27/04/2021 16:01	Arquivo CMAKE	1 KB
 pico_graphics.cpp	27/04/2021 17:57	Arquivo Fonte ...	10 KB
 pico_graphics.hpp	09/05/2021 11:30	Arquivo Fonte ...	3 KB
 pico_graphics_v2.cpp	26/04/2021 20:36	Arquivo Fonte ...	10 KB
 pico_graphics_v2.hpp	26/04/2021 20:34	Arquivo Fonte ...	3 KB
 README.md	23/02/2021 19:18	MD Documento	6 KB
 types.cpp	26/04/2021 20:07	Arquivo Fonte ...	2 KB

📁 > NVMe1TB (I:) > pico > rtc_psk_win2 > src_v2 > rv3028				
<input type="checkbox"/> Nome ^	Data de modificação	Tipo	Tamanho	
<input type="checkbox"/> CMakeLists.txt	27/04/2021 16:03	Documento de ...	1 KB	
📄 rv3028.cmake	23/02/2021 19:18	Arquivo CMAKE	1 KB	
📄 rv3028.cpp	10/05/2021 19:23	Arquivo Fonte ...	30 KB	
📄 rv3028.hpp	09/05/2021 18:47	Arquivo Fonte ...	13 KB	

📁 > NVMe1TB (I:) > pico > rtc_psk_win2 > src_v2 > st7789				
<input type="checkbox"/> Nome ^	Data de modificação	Tipo	Tamanho	
📄 CMakeLists.txt	27/04/2021 16:04	Documento de ...	1 KB	
📄 st7789.cmake	27/04/2021 16:05	Arquivo CMAKE	1 KB	
📄 st7789.cpp	04/05/2021 01:06	Arquivo Fonte ...	6 KB	
📄 st7789.hpp	04/05/2021 01:04	Arquivo Fonte ...	3 KB	

The subfolder shown below has been created by VSCode. It contains important files in which we can define (or alter) paths to our project folders or folders of libraries we use. In it we can also change parameters for the build process.





















^				
<input type="checkbox"/> Nome	Data de modificação	Tipo	Tamanho	
 c_cpp_properties.json	06/05/2021 00:05	Arquivo Fonte J...	1 KB	
 extensions.json	22/04/2021 16:22	Arquivo Fonte J...	1 KB	
 launch.json	11/05/2021 12:33	Arquivo Fonte J...	2 KB	
 settings.json	09/05/2021 11:50	Arquivo Fonte J...	2 KB	
 tasks.json	25/04/2021 11:29	Arquivo Fonte J...	1 KB	

I installed the pico-sdk folder on a 'branch' of the same disk root as my project:

📁 > NVMe1TB (I:) > pico > pico-sdk				
<input type="checkbox"/> Nome	Data de modificação	Tipo	Tamanho	
📁 .git	25/05/2021 00:01	Pasta de arquiv...		
📁 .github	25/05/2021 00:01	Pasta de arquiv...		
📁 cmake	25/05/2021 00:01	Pasta de arquiv...		
📁 docs	25/05/2021 00:01	Pasta de arquiv...		
📁 external	25/05/2021 00:01	Pasta de arquiv...		
📁 lib	25/05/2021 00:01	Pasta de arquiv...		
📁 src	25/05/2021 00:01	Pasta de arquiv...		
📁 test	25/05/2021 00:01	Pasta de arquiv...		
📁 tools	25/05/2021 00:01	Pasta de arquiv...		
📄 .gitignore	25/05/2021 00:01	Documento de ...	1 KB	
📄 .gitmodules	25/05/2021 00:01	Documento de ...	1 KB	
📄 CMakeLists.txt	25/05/2021 00:01	Documento de ...	2 KB	
📄 LICENSE.TXT	25/05/2021 00:01	Documento de ...	2 KB	
📄 pico_sdk_init.cmake	25/05/2021 00:01	Arquivo CMAKE	2 KB	
📄 pico_sdk_version.cmake	25/05/2021 00:01	Arquivo CMAKE	2 KB	
📄 README.md	25/05/2021 00:01	MD Documento	6 KB	

Also the pimoroni-pico repository I installed on the same 'branch' (I:\pico):

 > NVMe1TB (I:) > pico > pimoroni-pico				
<input type="checkbox"/> Nome	Data de modificação	Tipo	Tamanho	
 .git	22/05/2021 18:18	Pasta de arquiv...		
 .github	22/05/2021 18:18	Pasta de arquiv...		
 common	22/05/2021 18:18	Pasta de arquiv...		
 drivers	22/05/2021 18:18	Pasta de arquiv...		
 examples	22/05/2021 18:18	Pasta de arquiv...		
 libraries	22/05/2021 18:18	Pasta de arquiv...		
 micropython	22/05/2021 18:18	Pasta de arquiv...		
 .gitignore	22/05/2021 18:18	Documento de ...	1 KB	
 .gitmodules	22/05/2021 18:18	Documento de ...	1 KB	
 CMakeLists.txt	22/05/2021 18:18	Documento de ...	1 KB	
 LICENSE	22/05/2021 18:18	Arquivo	2 KB	
 pico_sdk_import.cmake	22/05/2021 18:18	Arquivo CMAKE	3 KB	
 pimoroni-pico-v0.2.1-micropyth...	22/05/2021 19:05	Arquivo UF2	945 KB	
 pimoroni-pico-v0.2.1-micropyth...	22/05/2021 19:04	Arquivo UF2	1 062 KB	
 README.md	22/05/2021 18:18	MD Documento	2 KB	
 setting-up-micropython.md	22/05/2021 18:18	MD Documento	2 KB	
 setting-up-the-pico-sdk.md	22/05/2021 18:18	MD Documento	3 KB	

File Explorer path: NVMe1TB (I:) > pico > pimoroni-pico > libraries >

<input type="checkbox"/> Nome	Data de modificação	Tipo	Tamanho
breakout_as7262	22/05/2021 18:18	Pasta de arquiv...	
breakout_colourlcd160x80	22/05/2021 18:18	Pasta de arquiv...	
breakout_colourlcd240x240	22/05/2021 18:18	Pasta de arquiv...	
breakout_dotmatrix	22/05/2021 18:18	Pasta de arquiv...	
breakout_encoder	22/05/2021 18:18	Pasta de arquiv...	
breakout_ioexpander	22/05/2021 18:18	Pasta de arquiv...	
breakout_ltr559	22/05/2021 18:18	Pasta de arquiv...	
breakout_matrix11x7	22/05/2021 18:18	Pasta de arquiv...	
breakout_mics6814	22/05/2021 18:18	Pasta de arquiv...	
breakout_msa301	22/05/2021 18:18	Pasta de arquiv...	
breakout_potentiometer	22/05/2021 18:18	Pasta de arquiv...	
breakout_rgbmatrix5x5	22/05/2021 18:18	Pasta de arquiv...	
breakout_roundlcd	22/05/2021 18:18	Pasta de arquiv...	
breakout_rtc	22/05/2021 18:18	Pasta de arquiv...	
breakout_sgp30	22/05/2021 18:18	Pasta de arquiv...	
breakout_trackball	22/05/2021 18:18	Pasta de arquiv...	
pico_display	22/05/2021 18:18	Pasta de arquiv...	
<input checked="" type="checkbox"/> pico_explorer	22/05/2021 18:18	Pasta de arquiv...	
<input checked="" type="checkbox"/> pico_graphics	22/05/2021 18:18	Pasta de arquiv...	
pico_rgb_keypad	22/05/2021 18:18	Pasta de arquiv...	
pico_scroll	22/05/2021 18:18	Pasta de arquiv...	
pico_unicorn	22/05/2021 18:18	Pasta de arquiv...	
pico_wireless	22/05/2021 18:18	Pasta de arquiv...	
CMakeLists.txt	22/05/2021 18:18	Documento de ...	1 KB

From the Pimoroni 'libraries' subfolder I needed the following: 'pico-explorer' and 'pico-graphics' as marked in blue in the image on the left.

I copied these two subfolders to my project's src_v2 subfolder because I wanted to make certain changes.

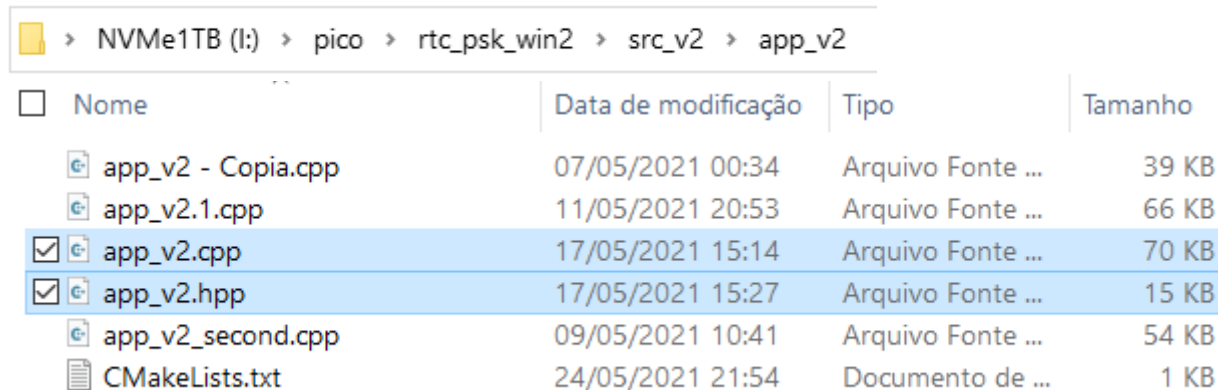
The Pimoroni 'drivers' I needed for this project are in subfolders of the 'drivers' subfolder shown below.

It are: 'rv3028' and 'st7789'.



<input type="checkbox"/> Nome	Data de modificação	Tipo	Tamanho
as7262	22/05/2021 18:18	Pasta de arquiv...	
esp32spi	22/05/2021 18:18	Pasta de arquiv...	
fatfs	22/05/2021 18:18	Pasta de arquiv...	
ioexpander	22/05/2021 18:18	Pasta de arquiv...	
is31fl3731	22/05/2021 18:18	Pasta de arquiv...	
ltp305	22/05/2021 18:18	Pasta de arquiv...	
ltr559	22/05/2021 18:18	Pasta de arquiv...	
msa301	22/05/2021 18:18	Pasta de arquiv...	
rv3028	22/05/2021 18:18	Pasta de arquiv...	
sdcard	22/05/2021 18:18	Pasta de arquiv...	
sgp30	22/05/2021 18:18	Pasta de arquiv...	
st7735	22/05/2021 18:18	Pasta de arquiv...	
st7789	22/05/2021 18:18	Pasta de arquiv...	
trackball	22/05/2021 18:18	Pasta de arquiv...	
vl531x	22/05/2021 18:18	Pasta de arquiv...	
CMakeLists.txt	22/05/2021 18:18	Documento de ...	1 KB

The projects main C++ file and C++ header file:



<input type="checkbox"/>	Nome	Data de modificação	Tipo	Tamanho
<input type="checkbox"/>	app_v2 - Copia.cpp	07/05/2021 00:34	Arquivo Fonte ...	39 KB
<input type="checkbox"/>	app_v2.1.cpp	11/05/2021 20:53	Arquivo Fonte ...	66 KB
<input checked="" type="checkbox"/>	app_v2.cpp	17/05/2021 15:14	Arquivo Fonte ...	70 KB
<input checked="" type="checkbox"/>	app_v2.hpp	17/05/2021 15:27	Arquivo Fonte ...	15 KB
<input type="checkbox"/>	app_v2_second.cpp	09/05/2021 10:41	Arquivo Fonte ...	54 KB
<input type="checkbox"/>	CMakeLists.txt	24/05/2021 21:54	Documento de ...	1 KB

The header file is defining various 'include' files. See the screenshot on the next page.

Some includes, especially those to header files of the pico-sdk I defined using a relative pointing-to indication like the following:

```
#include "../../../pico-sdk/src/common/pico_base/include/pico/types.h"
```

Because, for some time, I struggled with 'file not found' errors. Afterwards it became clear to me that this was caused by setup errors, even while I had the environment variable 'PICO-SDK-PATH' set.

The same I had to do with indicating the location of own project files, like this:

```
#include "../pico_explorer/pico_explorer.hpp"
```

```

rtc_psk_win2 > src_v2 > app_v2 > G+ app_v2.hpp > ...
1  #pragma once
2
3  #ifndef _RTC_PSK_NEW_HPP
4  #define _RTC_PSK_NEW_HPP
5  #include "../pico_explorer/pico_explorer.hpp"
6  #include <stdio.h>
7  #include <stdint.h>
8  // #include "../pico_graphics/pico_graphics.hpp"
9  #include "../rv3028/rv3028.hpp"
10 #include <float.h>
11 #include <string>
12 #include <vector>
13 #include <cstdint> // uint8_t etc.
14 #include <fstream>
15 #include <iomanip>
16 #include <iostream>
17
18 // #include "../pico-sdk/src/common/pico_util/include/pico/util/datetime.h"
19 #include "../pico-sdk/src/common/pico_base/include/pico/types.h"
20
21 using namespace pimoroni;
22
23 #define yy 0
24 #define mm 1
25 #define dd 2
26 #define wd 3
27 #define hh 4
28 #define mi 5
29 #define ss 6
30
31 const uint8_t USBPWR_PIN = 24;
32 const uint8_t LED_PIN = 25; // PICO_DEFAULT_LED_PIN; // Pin 25
33
34 const std::vector<int> ofs = {20, 60, 110, 180};
35 const std::vector<int> ofs3 = {5, 35, 65, 95, 125, 155, 185, 215, 235};
36 const std::vector<std::string> itmlst = {"ss", "mi", "hh", "wd", "dd", "mo", "yy", "rs", "mu", "??"};
37 const std::vector<std::string> itmlst_h = {"seconds", "minute", "hour", "weekday", "day", "month", "year", "reset", "menu", "help"};
38 const std::vector<std::string> wdays = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};
39 const std::vector<std::string> mnths = {"", "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};
40 const std::vector<std::string> settings = {"IsLED", "Is12HR", "IsALRM", "IsTIMR", "bGND", "fGND", "EXIT"};
41 const std::vector<std::string> settings_menu = {"led OnOff", "12/24 hrs", "set Alarm", "set Timer", "b_GND clr", "f_GND clr", "menu exit"};
42 const std::vector<std::string> colours = {"blk", "red", "aub", "ora", "yel", "grn", "blu", "whi"};
43 const std::vector<std::string> colours_h = {"black", "red", "aubergine", "orange", "yellow", "green", "blue", "white"};
44 // display background colour sets vectors of integer arrays
45 const std::vector<int> disp_black = {0, 0, 0}; // black see: https://rgbcolorcode.com/color/yellow
46 const std::vector<int> disp_red = {120, 0, 0}; // aubergine
47 const std::vector<int> disp_auber = {120, 40, 60}; // aubergine
48 const std::vector<int> disp_orange = {255, 128, 0}; // orange
49 const std::vector<int> disp_yellow = {255, 255, 0}; // yellow
50 const std::vector<int> disp_green = {0, 120, 0}; // green
51 const std::vector<int> disp_blue = {0, 0, 120}; // blue
52 const std::vector<int> disp_violet = {170, 0, 255}; // violet
53 const std::vector<int> disp_white = {255, 255, 255}; // white

```

The main C++ file: app_v2.cpp contains a list of #include's:

```
#include "app_v2.hpp"
#include "../pico_explorer/pico_explorer.hpp"
#include "../pico_graphics/font8_data.hpp"
// #include "../pico_explorer/pico_explorer.h"
#include "../rv3028/rv3028.hpp"

// #include <stdio.h>
#include <cstdio>
#include <cstring>
#include <stdint.h>
#include <iostream>
#include <bitset>
#include <sstream>
#include <cstdint> // uint8_t etc.
#include <iomanip>
// Next includes from pimoroni/pico-examples/explorer/demo.cpp
#include <string>
#include <string.h>
#include <math.h>
#include <vector>
#include <array>
#include <algorithm>
// #include <stdlib.h>
#include <float.h>

// #include "../pico_graphics/include/pico_graphics.hpp"
#include "../../../pico-sdk/src/common/pico_base/include/pico/types.h" // at the end of
#include "../../../pico-sdk/src/rp2_common/hardware_gpio/include/hardware/gpio.h"
#include "../../../pico-sdk/src/rp2_common/hardware_rtc/include/hardware/rtc.h"
#include "../../../pico-sdk/src/common/pico_util/include/pico/util/datetime.h"
#include "../../../pico-sdk/src/rp2_common/pico_platform/include/pico/platform.h" // h
#include "../../../pico-sdk/src/rp2_common/pico_unique_id/include/pico/unique_id.h"
#include "../../../pico-sdk/src/common/pico_stdlib/include/pico/stdlib.h"
#include "../../../micropython/py/vstr.c"
//
```


As we can read in the annotations in the top of the file app_v2.cpp, I added a few own created functions to the rv3028.cpp file:

```
/*  
Modifications by @paulsk, starting at 2021-04-05, for use with RPi Pico on a Pimoroni pico explorer base  
to make use of the Pimoroni external rv3028 rtc breakout, instead of the Pico's internal rtc).  
Note that I added the following functions to rv3028.hpp (main.hpp) and rtc.c (main.cpp):  
    bool rtc.get_time(stDateTime *t);  
    uint8_t rtc.yearday();  
    bool rtc.isleapyear(stDateTime *t);  
    uint8_t rtc.days_in_month(int i);  
*/
```















































The added functions are:

- rtc.get_time();
- rtc.yearday();
- rtc.isleapyear();
- rtc.days_in_month().

As Pimoroni created it's own namespace in their library files, I added near the top into app_v2.cpp the line:

```
using namespace pimoroni;
```

The project's main file `app_v2.cpp` contains 46 functions:

▼ OUTLINE	
 <code>a_itm_handler()</code>	 <code>not_yet(bool)</code>
 <code>a_menu_handler()</code>	 <code>pass()</code>
 <code>backp_to_EP()</code>	 <code>pico_explorer(buf) declaration</code>
 <code>btn_ab_itm(bool)</code>	 <code>pr_in_which_menu()</code>
 <code>btn_ab_settings(bool)</code>	 <code>pr_itmsel()</code>
 <code>btn_help()</code>	 <code>prep_datetime()</code>
 <code>btn_ID(uint8_t)</code>	 <code>reset_clk()</code>
 <code>btn_msg()</code>	 <code>reset_itm()</code>
 <code>btn_msgd(std::string, int)</code>	 <code>rest_fm_EP()</code>
 <code>btn_xy_itm(bool)</code>	 <code>set_12_24()</code>
 <code>btn_xy_settings(bool)</code>	 <code>set_alarm()</code>
 <code>ck_btn_press()</code>	 <code>set_background()</code>
 <code>ck_YN(std::string, std::string, std::string)</code>	 <code>set_BackupSwitchoverhMode(int)</code>
 <code>decr_incr_datetime(bool)</code>	 <code>set_blink()</code>
 <code>elapsedtime()</code>	 <code>set_datetime()</code>
 <code>fail_msgs(uint8_t)</code>	 <code>set_DispColour(bool, int, bool)</code>
 <code>get_alarm()</code>	 <code>set_foreground()</code>
 <code>get_datetime()</code>	 <code>set_timer()</code>
 <code>hexify(const buffer &)</code>	 <code>setup()</code>
 <code>hsv_to_rgb(rgb_t *)</code>	 <code>upd_app_time()</code>
 <code>led_toggle()</code>	 <code>USBpwr()</code>
 <code>main()</code>	 <code>w_btn()</code>
 <code>my_rgb(int)</code>	 <code>z main loop()</code>

The projects main header file: app_v2.hpp defines vector arrays: itmlst, itmlst_h, wdays, mnths, settings, settings_menu.

Colours and colours_h. It also defines rgb colours for the lcd, from: disp_black to disp_white.

```
#define yy 0
#define mm 1
#define dd 2
#define wd 3
#define hh 4
#define mi 5
#define ss 6

const uint8_t USBPWR_PIN = 24;
const uint8_t LED_PIN = 25; // PICO_DEFAULT_LED_PIN; // Pin 25

const std::vector<int> ofs = {20, 60, 110, 180};
const std::vector<int> ofs3 = {5,35,65,95,125,155,185,215, 235};
const std::vector<std::string> itmlst = {"ss", "mi", "hh", "wd", "dd", "mo", "yy", "rs", "mu", "??"};
const std::vector<std::string> itmlst_h = {"seconds", "minute", "hour", "weekday", "day", "month", "year", "reset", "menu", "help"};
const std::vector<std::string> wdays = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};
const std::vector<std::string> mnths = {"", "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};
const std::vector<std::string> settings = {"IsLED", "Is12HR", "IsALRM", "IsTIMR", "bGND", "fGND", "EXIT"};
const std::vector<std::string> settings_menu = {"led OnOff", "12/24 hrs", "set Alarm", "set Timer", "b_GND clr", "f_GND clr", "menu exit"};
const std::vector<std::string> colours = { "blk", "red", "aub", "ora", "yel", "grn", "blu", "whi"};
const std::vector<std::string> colours_h = { "black", "red", "aubergine", "orange", "yellow", "green", "blue", "white"};
// display background colour sets vectors of integer arrays
const std::vector<int> disp_black = {0, 0, 0}; // black see: https://rgbcolorcode.com/color/yellow
const std::vector<int> disp_red = {120, 0, 0}; // aubergine
const std::vector<int> disp_auber = {120, 40, 60}; // aubergine
const std::vector<int> disp_orange = {255, 128, 0}; // orange
const std::vector<int> disp_yellow = {255, 255, 0}; // yellow
const std::vector<int> disp_green = {0, 120, 0}; // green
const std::vector<int> disp_blue = {0, 0, 120}; // blue
const std::vector<int> disp_violet = {170, 0, 255}; // violet
const std::vector<int> disp_white = {255, 255, 255}; // white
```

The project's main header file defines
six enumerated lists:

```
enum menu_order
{
    MENU_MAIN,
    MENU_SETTINGS,
    MENU_BACK
};

enum menu_settings_order
{
    SETT_LED,           // 0
    SETT_12HR,          // 1
    SETT_ALARM,          // 2
    SETT_TIMR,           // 3
    SETT_BGND,           // 4
    SETT_FGND,           // 5
    SETT_BACK            // 6
};

enum menu_itm_order
{
    ITM_SECONDS,        // 0
    ITM_MINUTES,         // 1
    ITM_HOURS,           // 2
    ITM_WEEKDAY,         // 3
    ITM_DATE,            // 4
    ITM_MONTH,           // 5
    ITM_YEAR,            // 6
    ITM_RESET,           // 7
    ITM_MENU,            // 8
    ITM_HELP             // 9
};
```

```
enum disp_colour_order
{
    DISP_BLACK,          // 0
    DISP_RED,             // 1
    DISP_AUBERGINE,       // 2
    DISP_ORANGE,          // 3
    DISP_YELLOW,          // 4
    DISP_GREEN,           // 5
    DISP_BLUE,            // 6
    DISP_WHITE            // 7
};

enum z_main_loop_order
{
    DOIT_WEEKDAY,        // 0
    DOIT_MONTH,           // 1
    DOIT_YEAR,            // 2
    DOIT_TIME             // 3
};

enum setup_call_order
{
    SU_RTC_INIT,          // 0 0x01
    SU_RTC_SETUP,         // 1 0x02
    SU_REST_FM_INIT_LED,  // 2 0x04
    SU_REST_FM_INIT_BF,   // 3 0x08
    SU_RTC_UPDATETIME,    // 4 0x20
    SU_UPD_APP_TIME,      // 5 0x40
    SU_SUCCESSFULL,       // 6
    SU_FAILED,            // 7
    SU_UNKNOWN            // 8
};
```

Then app_v2.hpp defines the following variables:

```
int menu = MENU_MAIN; // Set to use the main
int itm = ITM_HELP; // Set start value of
rs, 8 = ?? (help)
int sett_itm = SETT_LED; // idem for setting
int bgnd_colour = DISP_ORANGE; // Global dis
int fgnd_colour = DISP_YELLOW; // Global d
int p_x_default = 15;
int usr_dat_len_at_backup = 0; // Holds the
rest_fm_EP() )
int app_time_IsPM = -1;

char datetime_buf[256];
char *datetime_str = &datetime_buf[0];
```

Finally, app_v2.hpp defines two other vector arrays and three type definitions, one of a buffer vector array and two structures:

```
#ifndef TIME_ARRAY_LENGTH
#define TIME_ARRAY_LENGTH 7 // Total number of writable values in device
#endif

std::vector<int> app_time = { 0, 0, 12, 5, 1, 5, 2021}; // {secs, mins, hrs, weekday, date, month, year}
std::vector<int> v_settings = { 1, 1, 0, 0, 0, 0, 0}; // IsLED, Is12, IsAM, IsALRM, IsTIMR, fGND, bGND

// See: https://codereview.stackexchange.com/questions/242052/conversion-into-hexadecimal-using-c , by by @Malvineous
typedef std::vector<uint8_t> buffer;

// We need to make a custom type so we can control which function the compiler will call.
struct hexbuffer {
    const buffer& innerbuf;
};

//-----

typedef struct {
    float r;
    float g;
    float b;
} rgb_t;
```

For the rest the app_v2.hpp contains a prototype definition for each function in app_v2.cpp.

The project's main C++ file contains the following global objects:

- pico_explorer;
- rtc.

other variables like boolean flags: lStDT, t_chgd, lastHrOfDay, ledIsOn, lUSB, lRTC, and the following global integer variables: btns, rgb_h, Ut and UltItmChgdTime.

```
uint16_t buf[PicoExplorer::WIDTH * PicoExplorer::HEIGHT];  
PicoExplorer pico_explorer(buf); // create an instance of the PicoExplorer object  
  
pico_unique_board_id_t p_ID;  
  
bool lStDT = true;  
bool t_chgd = false; // See btn funcs  
bool lastHrOfDay = false;  
int btns = 0;  
bool ledIsOn = false;  
int rgb_h = 0;  
bool lUSB;  
bool lRTC = false;  
int Ut = 0;  
int UltItmChgdTime = 0;  
RV3028 rtc; // Create instance of RV3028 object
```

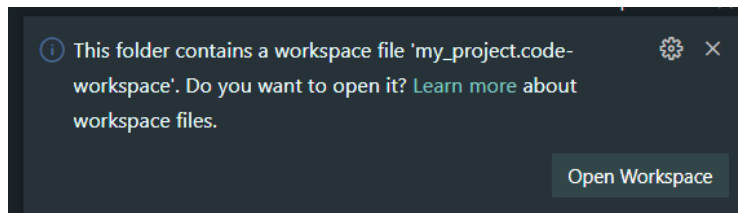
The build and flash process

I start a VSCode session as follows:

- from the search bar in the MS Windows 10 desktop I type “ub” which then will show the app. See the image right top. After starting it, there will popup a screen like the image right down. Then I issue the ‘cd’ command to the project’s root folder, in my case:

`/mnt/i/pico/rtc_psk_win2`. Arrived there I issue the command `code .`

(`code<space><decimal_point>`). Which will start a VSCode session. After VSCode has started, it usually will popup a window in the lower right of its window:



My 'reply' to this popup is always that I click on the button: **'Open Workspace'**.

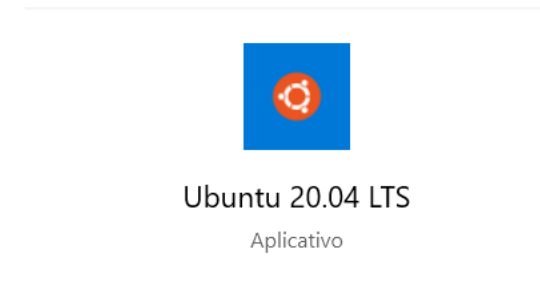
I built the project from within the VSCode session, from within it's terminal window. If a terminal window is not open, I open it via the **'View'** menu > **Terminal** (Ctrl-ç)

During the first development session, I had to create a **'build'** folder. Then I set the current folder to the build folder, in this project: `/mnt/i/pico/rtc_psk_win2/build/`.

If it is not our first build session we can opt to clean the contents of the build subfolder with the command `rm -rf *`. Warning: be careful with this powerfull (and destructive) command. Check that you are in the correct subfolder before issuing this command!

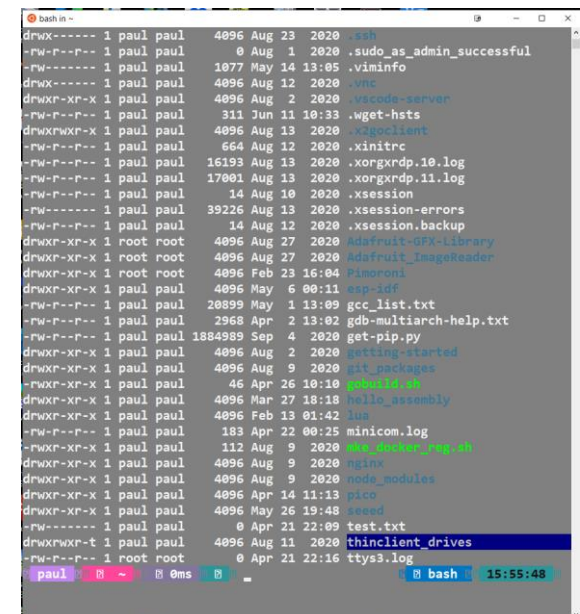
1st step: `cmake ../src_v2/`

When this step has completed without error:



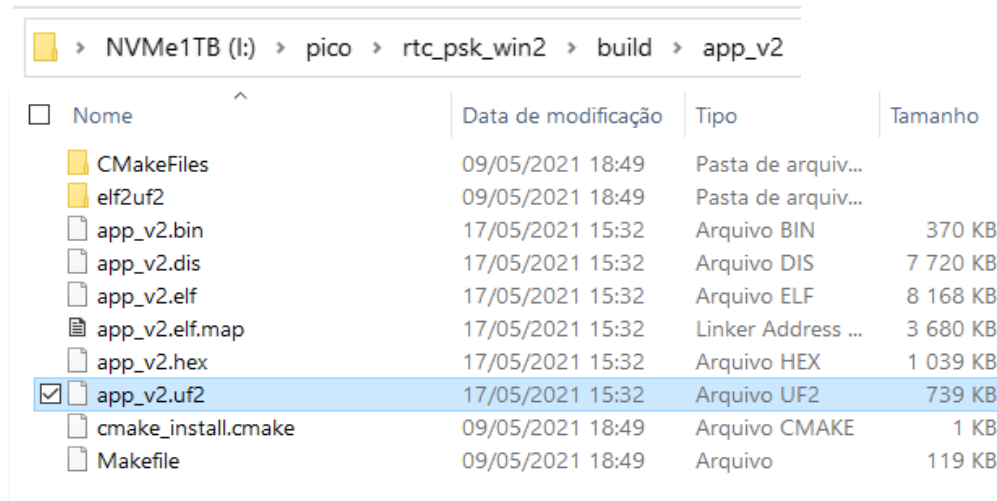
Abrir

Executar como administrador



2nd step: 'make'.

3rd step: When the second step also has been completed without error, the resulting .uf2 file (that we need to flash into a Raspberry Pi Pico), is located in the subfolder:



Nome	Data de modificação	Tipo	Tamanho
CMakeFiles	09/05/2021 18:49	Pasta de arquiv...	
elf2uf2	09/05/2021 18:49	Pasta de arquiv...	
app_v2.bin	17/05/2021 15:32	Arquivo BIN	370 KB
app_v2.dis	17/05/2021 15:32	Arquivo DIS	7 720 KB
app_v2.elf	17/05/2021 15:32	Arquivo ELF	8 168 KB
app_v2.elf.map	17/05/2021 15:32	Linker Address ...	3 680 KB
app_v2.hex	17/05/2021 15:32	Arquivo HEX	1 039 KB
<input checked="" type="checkbox"/> app_v2.uf2	17/05/2021 15:32	Arquivo UF2	739 KB
cmake_install.cmake	09/05/2021 18:49	Arquivo CMAKE	1 KB
Makefile	09/05/2021 18:49	Arquivo	119 KB

Finally this file 'app_v2.uf2' has to be flashed into a RPi Pico by putting the RPi Pico into '[BootSEL](#)' mode and copy the file '[app_v2.uf2](#)' into the '[RPI-RP2](#)' flash 'disk' folder. When the flash process has completed, the RPi Pico will reboot and the RTC app will come to life.

In the main file, app_v2.cpp, I have programmed quite a bit of print statements in the form "std::cout << ...". This is done so I could monitor the state of the program run cycle (see the screenshot below).

```

Loop nr: 993. Elapsed time: 123 seconds. In minutes: 2
We are in menu: main
t: 16:43:20
upd_app_time(): the hours value of the rtc is: 16
upd_app_time(): the result of rtc.is12hour() is: 0
upd app time(): app time vector array is set to: 21-6-11, weekday: 4, t: 16:43:21

```

After pressing the 'reset' button (if a reset button is present on your Pico) the following monitor text will be presented:

```
-----  
setup(): RTC online! hardware ID: '51', in hex: '0x33 '. Part. ID: '3', Version: '3'  
rest_fm_EP(): entering...  
rest_fm_EP(): the LED On/Off value retrieved from register # 0x1f is: '0x00 '  
rest_fm_EP(): v_settings.at(SETT_LED) has been set to: 0  
rest_fm_EP(): usr_ram2_val (LCD back- and foreground colour data) retrieved from register # 0x20 = '0x36 '  
rest_fm_EP(): usr_ram2_val & 0x0f = '54' & '15' = 3= orange  
rest_fm_EP(): usr_ram2_val & 0xf0 = '54' & '240' = 6= blue  
rest_fm_EP(): the LCD background colour value retrieved from EEPROM USR_RAM2 is: 6  
rest_fm_EP(): the LCD foreground colour value retrieved from EEPROM USR_RAM2 is: 3  
rest_fm_EP(): leaving with return value: 0  
setup(): result received from call to rest_fm_EP() is: 0  
upd_app_time(): the hours value of the rtc is: 16  
upd_app_time(): the result of rtc.is12hour() is: 0  
|  
upd_app_time(): app_time vector array is set to: 21-6-11, weekday: 4, t: 16:47:8  
setup(): Read/Write Time - RTC Example  
setup(): leaving...  
Real-time clock test.  
Current datetime stamp is: 11/06/2021.  
-----  
Btns: A: /\ Incr, B: \/ Decr, <- X: Next, -> Y: Prev  
-----
```

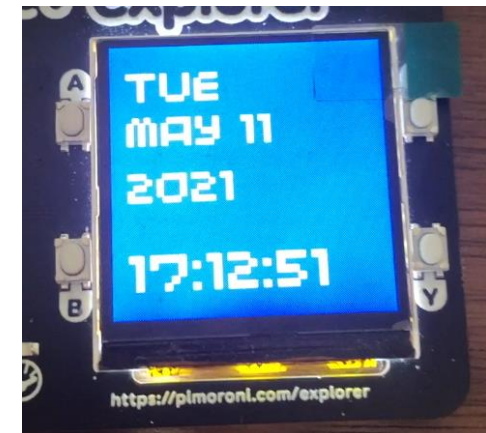
How the program works

Note: when, in this text is mentioned 'realtime clock' or 'rtc', the realtime clock of the Pimoroni RV3028 breakout board is meant, not the built-in rtc of the RPi Pico.

In this project I created two functions to save and retrieve a few general settings:

- the foreground and background colour of the lcd;
- the status of the built-in led blinking mode: blinking or not blinking.

These settings are saved into and retrieved from two bytes that are available as USER_RAM1 and USER_RAM2. See the functions: `backp_to_EP()` and `rest_fm_EP()` for the details.



The date and time values retrieved from the realtime clock are saved into a vector array named: 'app_time'. See the functions: 'set_datetime()' and 'upd_app_time()'. The latter function first calls the rtc's function: `rtc.updateTime()`, then 'fills' the `app_time` array with the most recent date and time values from the rtc. The function gets also the status of the 12/24 clock mode. When in 12 Hour mode, the 'am' or 'pm' will be shown on the lcd.

Buttons

The project makes use of the four buttons ('A', 'B', 'X' and 'Y') that are on the pico explorer base, 2 x 2 on both sides of the lcd.

The buttons have been given the following functions:

Button	Function #1	Function #2
A	Increase (yy, mm, ..)	
B	Decrease (idem)	
X	Up (menu item)	No (e.g.: reset clock? Or blink led?)
Y	Dn (menu item)	Yes

The menus

There exist a 'main menu' and a 'settings menu'. The following items are the main menu items:

- help;
- minute;
- hour;
- weekday;
- day;
- month;
- year;
- reset;
- menu;
-

The default menu-item is 'help'.

When in 'help' pressing either 'A' or 'B' button will display the help page.

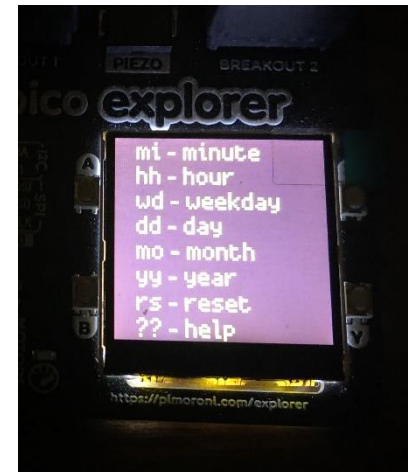
When the user has chosen before another menu-item than 'help', after 30 seconds, the menu-item selection will revert to 'help'.

The main menu item 'menu' brings the program into the 'settings menu'.

To get into the settings menu the user has first to select the 'menu' item in the main menu, then press 'A' or 'B'. The program will then display for one second the text 'MENU SETTINGS'

The 'settings menu' has the following items:

- menu exit;
- f_gnd; (to change the colour of the foreground (the text colour);
- b_gnd (to change the colour of the background);
- set timer;
- set alarm; (not implemented yet)



- 12/24 HRS;
- led onoff.

For the following two settings the program will ask the user for confirmation:

'Reset clock?	'Blink LED?
BTN X = no	BTN X = no
BTN Y = yes'	BTN Y = yes'

Note that each time the user presses a button a message will be shown on the lcd. The user has to wait pressing any button until the notification text has disappeared.

Because the realtime clock break module has a backup battery (no rechargeable !), the clock will keep the time, even when the raspberry pi is not powered via usb or another external electrical source, e.g.: a battery. The software sets the external realtime clock module to detect when the RPi Pico is without of electrical power.

The reset clock function set the clock to: Saturday, May 1, 2021, 12:00:00.

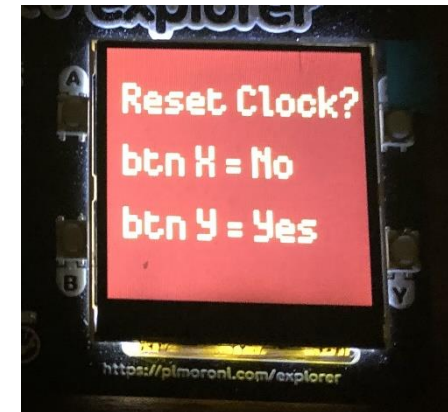
The SET-TIMER function is shown in the image on the right:

At the end of the timer countdown there will appear a flashing screen as shown in the image below:



However the Pimoroni pico explorer base consists of a tiny speaker, my experience is that the volume of this speaker is too soft. That is why I did not add it to the timer alarm moment.

ToDo: add an audio output to one of the PWM pins of the Pico to drive a more audible speaker.



This ends the report of my project RPi_Pico_cpp_ext_rtc.

Thank you for taking your time to read it.

Finally my thanks for the support of various people that I asked help for on my 'journey' into building C++ apps for the RPi Pico. A special word of thanks for the support of Constantin Koch.

Feedback always appreciated.

Paulus Schulinck,
Lisbon, Portugal
June 11, 2021

@PaulskPt on GitHub;
@paulsk on Discord, CircuitPython (member of 'deepdivers');
@ct7agr on Twitter.

ⁱ <https://github.com/pimoroni/i2cdevice-python>

ⁱⁱ <https://github.com/pimoroni/rv3028-python>

ⁱⁱⁱ <https://shop.pimoroni.com/products/raspberry-pi-pico?variant=32402092294227>

^{iv} <https://shop.pimoroni.com/products/pico-explorer-base>

^v <https://shop.pimoroni.com/products/rv3028-real-time-clock-rtc-breakout>

^{vi} <https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/3>

^{vii} <https://github.com/raspberrypi/pico-sdk>

^{viii} https://github.com/constiko/RV-3028_C7-Arduino_Library

^{ix} : <https://github.com/pimoroni/pimoroni-pico>

^x <https://code.visualstudio.com/>

^{xi} <https://docs.microsoft.com/en-us/windows/wsl/install-win10>

^{xii} <https://datasheets.raspberrypi.org/pico/getting-started-with-pico.pdf>

^{xiii} <https://datasheets.raspberrypi.org/pico/raspberry-pi-pico-c-sdk.pdf>