**********************Draft***********************

Pico_w MQTT Client

Mosquitto or Ultibo QEMU or Ultibo Hardware RPi3 Broker

GPIO Control

Adding a 2nd socket to Pico_W freertos iperf

To provide Debug information previously provided by a hard wired connection to the Pico_W UART

Now two version of picow_freertos_iperf.c.ntp & picow_freertos_iperf.c.rtc

ntp is used picow_freertos_iperf.c

or

pub_time is used picow_freertos_iperf.c.rtc is copied to picow_freertos_iperf.c

latest changes have not been made to picow_freertos_iperf.c.rtc

04/28/23

**********************Draft***********************

The following ver 0.0.04 is using picow_freertos_iperf.c

From: Christoph M. Wintersteiger
Subject: Re: [lwip-users] lwip on pico_w
Date: Thu, 27 Apr 2023 22:17:22 +0100

…

```
cyw43_arch_lwip_begin();

err_t r = mqtt_publish(client, topic, payload, payload_length, qos, retain, cb, arg);

cyw43_arch_lwip_end();
```

```
devel@pi4-50:~/pico_w-mqtt/pico_w/wifi/freertos/iperf $ grep "cyw43_arch_lwip_begin()" picow_freertos_iperf.c | wc
     12      12     347
devel@pi4-50:~/pico_w-mqtt/pico_w/wifi/freertos/iperf $ grep "cyw43_arch_lwip_end()" picow_freertos_iperf.c | wc
     12      12     315
```

were added to picow_freertos_iperf.c

```
devel@pi4-50:~/pico_w-mqtt/remote1 $ ./exe-ocd.sh
Open On-Chip Debugger 0.11.0-g228ede4-dirty (2022-08-27-19:45)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
adapter speed: 1000 kHz

Info : Hardware thread awareness created
Info : Hardware thread awareness created
Info : RP2040 Flash Bank Command
Info : BCM2835 GPIO JTAG/SWD bitbang driver
Info : clock speed 1001 kHz
        .
        .
        .
```

```
    xPSR: 0x01000000 pc: 0x00000138 msp: 0x20041f00
    ** Programming Finished **
    ** Verify Started **
    target halted due to debug-request, current mode: Thread
    xPSR: 0x01000000 pc: 0x00000138 msp: 0x20041f00
    target halted due to debug-request, current mode: Thread
    xPSR: 0x01000000 pc: 0x00000138 msp: 0x20041f00
    ** Verified OK **
    ** Resetting Target **
    shutdown command invoked

    Starting FreeRTOS on core 0:
    Connected.
    MQTT client "remote1" connection cb: status 0
    MQTT client "remote1" request cb: err 0
    MQTT client "remote1" request cb: err 0
    Starting server at 192.168.1.176 on port 4001
    ntp_task starts
    ntp address 72.30.35.89
    0x200220d8 0x200220fc 0x200220f4
    Sunday 24 April 14:46:11 2023      got ntp response: 24/04/2023 14:46:10

    ../pi_tcp_tests/cli1
    Socket created successfully
    Connected with server successfully
    Starting FreeRTOS on core 0: ver 0.0.04 remote1
    Connecting to Wi-Fi...Connected. iperf server 192.168.1.176 4001  starting watchdog timer task
mqtt_ip = 0xd401a8c0 mqtt_port = 1883  mqtt_connect 0x0 mq_con 0x1 mqtt_connect 0x1 mq_con
0x1     24/04/2023 14:48:35
    mqtt_connect 0x1 mq_con 0x1 mqtt_connect 0x1 mq_con 0x1 mqtt_connect 0x1 mq_con 0x1
mqtt_connect 0x1 mq_con 0x1 123`�o�
```

if line 375 is changed from vTaskDelay(10000); to vTaskDelay(100); the *** PANIC *** will occur fairly quit after start up.

```
git diff runtime.c
diff --git a/src/rp2_common/pico_runtime/runtime.c b/src/rp2_common/pico_runtime/runtime.c
index f9018d0..eee8296 100644
--- a/src/rp2_common/pico_runtime/runtime.c
+++ b/src/rp2_common/pico_runtime/runtime.c
@@ -275,7 +275,9 @@ void __assert_func(const char *file, int line, const char *func, const char
*fai
 void __attribute__((noreturn)) panic_unsupported() {
     panic("not supported");
 }
-
+void PICO_PANIC_FUNCTION(void) {
+    printf("my attempt at picp_panic_function\n");
+}
 // PICO_CONFIG: PICO_PANIC_FUNCTION, Name of a function to use in place of the stock
panic function or empty string to simply breakpoint on panic, group=pico_runtime
 // note the default is not "panic" it is undefined
 #ifdef PICO_PANIC_FUNCTION
```

*** PANIC ***

pcb->snd_queuelen >= pbuf_clen(next->p)

Mosquitto or Ultibo QEMU or Hardware RPi3 Broker
main_task
watchdog_task
mqtt_task
gpio_task
rtc_task
socket_task
blink_task

Build Steps
RTC time setting

Mosquitto Broker
GPIO

Build Steps
“git clone https://github.com/develone/pico_w-mqtt.git -b dev”
“cd pico_w-mqtt”
Modify the script “6remotes.sh” WIFI_SSID with your SSID and WIFI_PASSWORD with your PASSWORD.
Modify the file “pico_w/wifi/freertos/iperf/picow_freertos_iperf.c” WIFI_PASSWORD with your PASSWORD.
Depending on which broker is going to be used the there are several parameters that can be changed (port and ip of broker Mosquitto & Ultibo hardware port 1883 Ultibo QEMU 18830)

```
 /*192.168.1.212 0xc0a801d4 LWIP_MQTT_EXAMPLE_IPADDR_INIT pi4-50*/
                              #define LWIP_MQTT_EXAMPLE_IPADDR_INIT = 
IPADDR4_INIT(PP_HTONL(0xc0a801d4))

/*192.168.1.231 0xc0a801d4 LWIP_MQTT_EXAMPLE_IPADDR_INIT ultibo*/
                              #define LWIP_MQTT_EXAMPLE_IPADDR_INIT = 
IPADDR4_INIT(PP_HTONL(0xc0a801e7))
```

“./6remotes.sh” creates 6 copies of the program “remotex/pico_w/wifi/freertos/iperf/picow_freertos_iperf_server_mqtt.elf” each with a different hostname. In addition copies “exe-ocd.sh” to each of the six folders remotex.
“exe-ocd.sh” uses openocd to program the Pico_W

```
#!/bin/bash
openocd -f interface/raspberrypi-swd.cfg -f target/rp2040.cfg -c "program 
pico_w/wifi/freertos/iperf/picow_freertos_iperf_server_mqtt.elf verify reset exit"
```

It also runs the script “build_cli.sh”. The IP of the pico_w remotes in the file client.c on your network need to modified to your network.

The script “build_cli.sh” creates 6 programs (cli1, cli2, cli3, cli4, cli5, and cli6) in the folder pi_tcp_tests.

```
#!/bin/bash
```

```
cd pi_tcp_tests

rm -f cli1 cli2 cli5 cli6

gcc -v client.c -Drem1  -o cli1
gcc -v client.c -Drem2  -o cli2
gcc -v client.c -Drem3  -o cli3
gcc -v client.c -Drem4  -o cli4

gcc -v client.c -Drem5  -o cli5
gcc -v client.c -Drem6  -o cli6
```

https://github.com/develone/Tools/blob/master/Installer/Core/Linux/ultiboinstaller.sh can be use to installed Lazarus IDE (Ultibo Edition) on RPi . The script requires

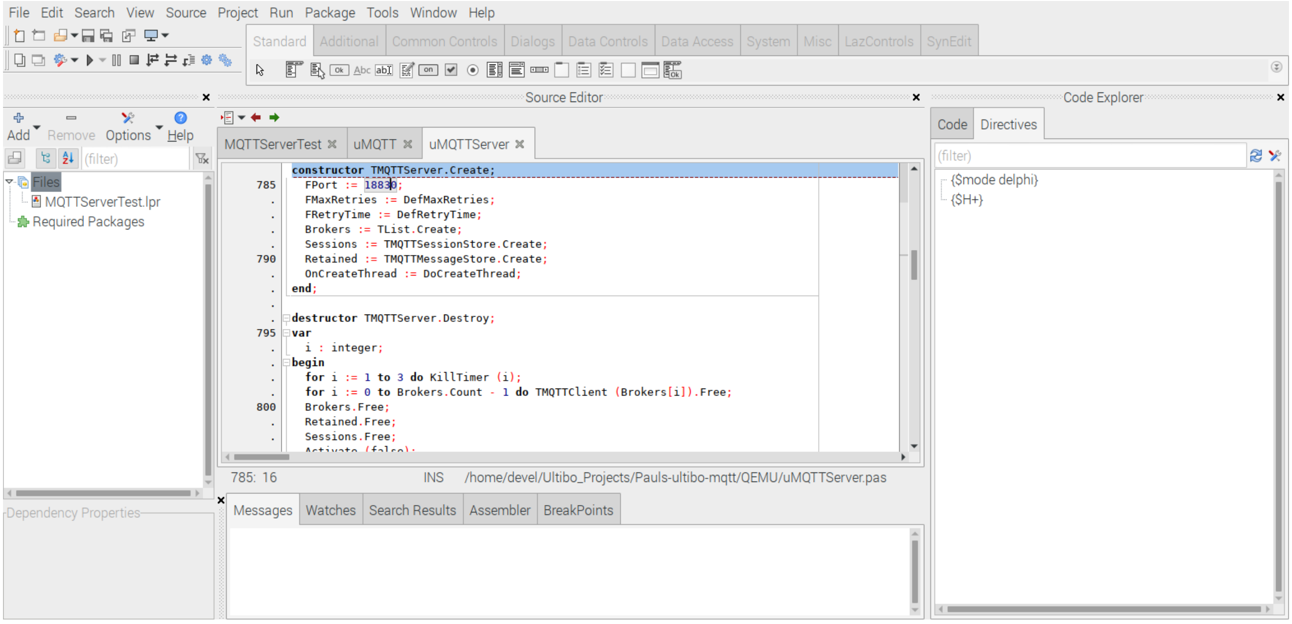
```
 requirePackage "libgtk2.0-dev"
 requirePackage "libcairo2-dev"
 requirePackage "libpango1.0-dev"
 requirePackage "libgdk-pixbuf2.0-dev"
 requirePackage "libatk1.0-dev"
requirePackage "libghc-x11-dev"
```

git clone https://github.com/develone/Ultibo_Projects.git

The IP of the host needs to match your host system virtual bare metal RPi

```
          MQC.Host := '192.168.1.212';
          MQC.Username := 'testuser';
          MQC.Password := 'password123';
          MQC.LocalBounce := false;
          MQC.Activate (true);
          MQC.Publish ('pub_time', #0#11'hello there', qtEXACTLY_ONCE, false);

        end;
```

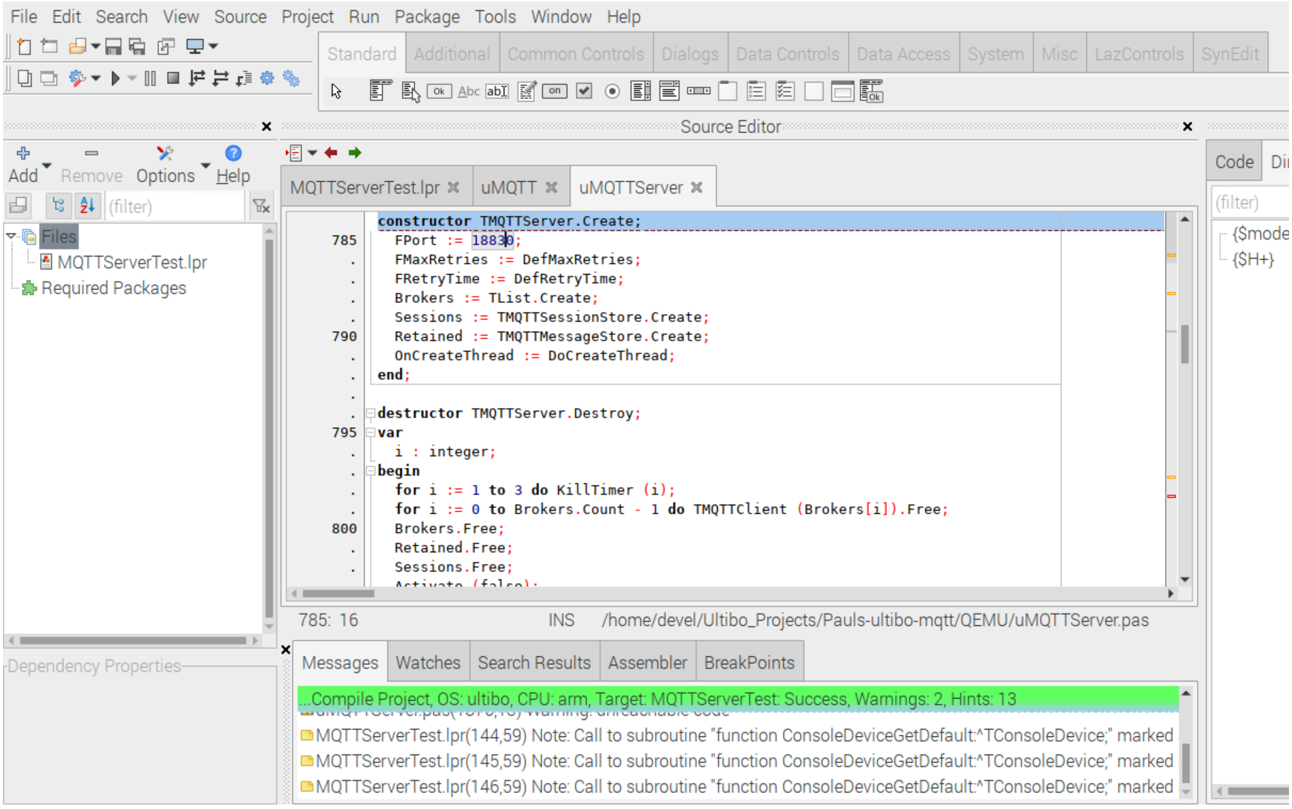

File Edit Search View Source Project Run Package Tools Window Help
Standard Additional Common Controls Dialogs Data Controls Data Access System Misc LazControls SynEdit
Source Editor
Code Explorer
Add Remove Options Help
(filter)
Files
MQTTServerTest.lpr
Required Packages
MQTTServerTest uMQTT uMQTTServer
Code Directives
{$mode delphi}
{$H+}
Dependency Properties
Messages Watches Search Results Assembler BreakPoints

```
constructor TMQTTServer.Create;
  FPort := 18830;
  FMaxRetries := DefMaxRetries;
  FRetryTime := DefRetryTime;
  Brokers := TList.Create;
  Sessions := TMQTTSessionStore.Create;
  Retained := TMQTTMessageStore.Create;
  OnCreateThread := DoCreateThread;
end;

destructor TMQTTServer.Destroy;
var
  i : integer;
begin
  for i := 1 to 3 do KillTimer (i);
  for i := 0 to Brokers.Count - 1 do TMQTTClient (Brokers[i]).Free;
  Brokers.Free;
  Retained.Free;
  Sessions.Free;
  Activate (false);
```

785: 16   INS   /home/devel/Ultibo_Projects/Pauls-ultibo-mqtt/QEMU/uMQTTServer.pas

Depressing the Run/Compile or Run/Cleanup and Build



```
constructor TMQTTServer.Create;
  FPort := 18830;
  FMaxRetries := DefMaxRetries;
  FRetryTime := DefRetryTime;
  Brokers := TList.Create;
  Sessions := TMQTTSessionStore.Create;
  Retained := TMQTTMessageStore.Create;
  OnCreateThread := DoCreateThread;
end;

destructor TMQTTServer.Destroy;
var
  i : integer;
begin
  for i := 1 to 3 do KillTimer (i);
  for i := 0 to Brokers.Count - 1 do TMQTTClient (Brokers[i]).Free;
  Brokers.Free;
  Retained.Free;
  Sessions.Free;
  Activate (false);
```

```
785: 16    INS    /home/devel/Ultibo_Projects/Pauls-ultibo-mqtt/QEMU/uMQTTServer.pas
```

```
...Compile Project, OS: ultibo, CPU: arm, Target: MQTTServerTest: Success, Warnings: 2, Hints: 13
MQTTServerTest.lpr(144,59) Note: Call to subroutine "function ConsoleDeviceGetDefault:^TConsoleDevice;" marked
MQTTServerTest.lpr(145,59) Note: Call to subroutine "function ConsoleDeviceGetDefault:^TConsoleDevice;" marked
MQTTServerTest.lpr(146,59) Note: Call to subroutine "function ConsoleDeviceGetDefault:^TConsoleDevice;" marked
```

When the green bar appears the project is ready to run as virtual bare metal RPi.

Mosquitto Broker

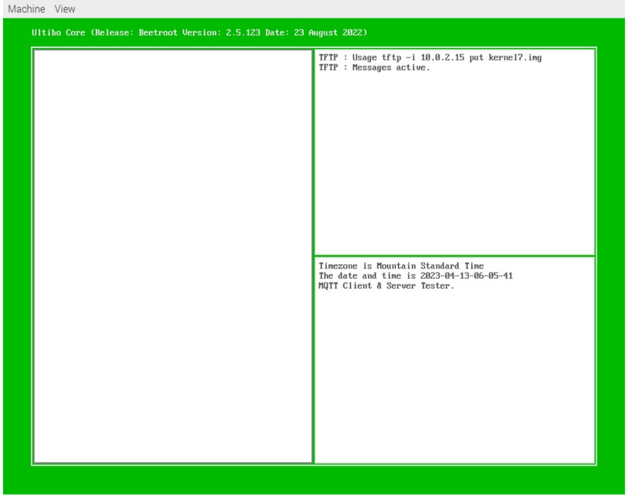
```
diff /usr/share/doc/mosquitto/examples/mosquitto.conf /etc/mosquitto/mosquitto.conf
512c512,522
< #allow_anonymous false
---
> #listener  8883 192.168.1.211
> #listener 1884 192.168.1.211
> listener 9883
> #listener 9883 192.168.1.175
> listener 1883
> user testuser
> per_listener_settings true
> #password_file /etc/mosqitto/mosquitto-pw
> password_file /home/devel/mosquitto-pw
> #acl_file file /etc/mosquitto/acl_file.conf
> allow_anonymous false
513a524
> #log_dest stdout
```

```
mosquitto -c /etc/mosquitto/mosquitto.conf
mosquitto_sub -t 'update/memo' -u 'testuser' -P 'password123'
mosquitto_sub -h pi4-60 -p 1883 -t 'update/memo' -u 'testuser' -P 'password123'
```

Ultibo QEMU “https://github.com/develone/Ultibo_Projects/tree/master/Pauls-ultibo-mqtt” u16_t mqtt_port = 18830; instead of default u16_t mqtt_port = 1883;

```
devel@pi4-50:~/Ultibo_Projects/Pauls-ultibo-mqtt/QEMU $ ./startqemu.sh
#!/bin/bash
qemu-system-arm -machine versatilepb -cpu cortex-a8 -kernel kernel.bin \
-net
user,hostfwd=tcp::5080-:80,hostfwd=tcp::5023-:23,hostfwd=tcp::18830-:18830,hostfwd=udp::5069
-:69,hostfwd=tcp::6050-:5050 -net nic \
 -drive file=disk.img,if=sd,format=raw
```

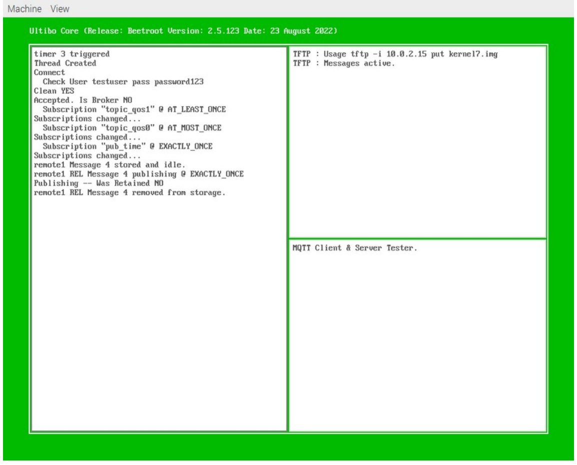
Starts a 3 pane Window. First step would be depressing '5' : MQ.Activate (true);



```
Ultibo Core (Release: Beetroot Version: 2.5.123 Date: 23 August 2022)

TFTP : Usage tftp -i 10.0.2.15 put kernel7.img
TFTP : Messages active.

Timezone is Mountain Standard Time
The date and time is 2023-04-13-06-05-41
MQTT Client & Server Tester.
```



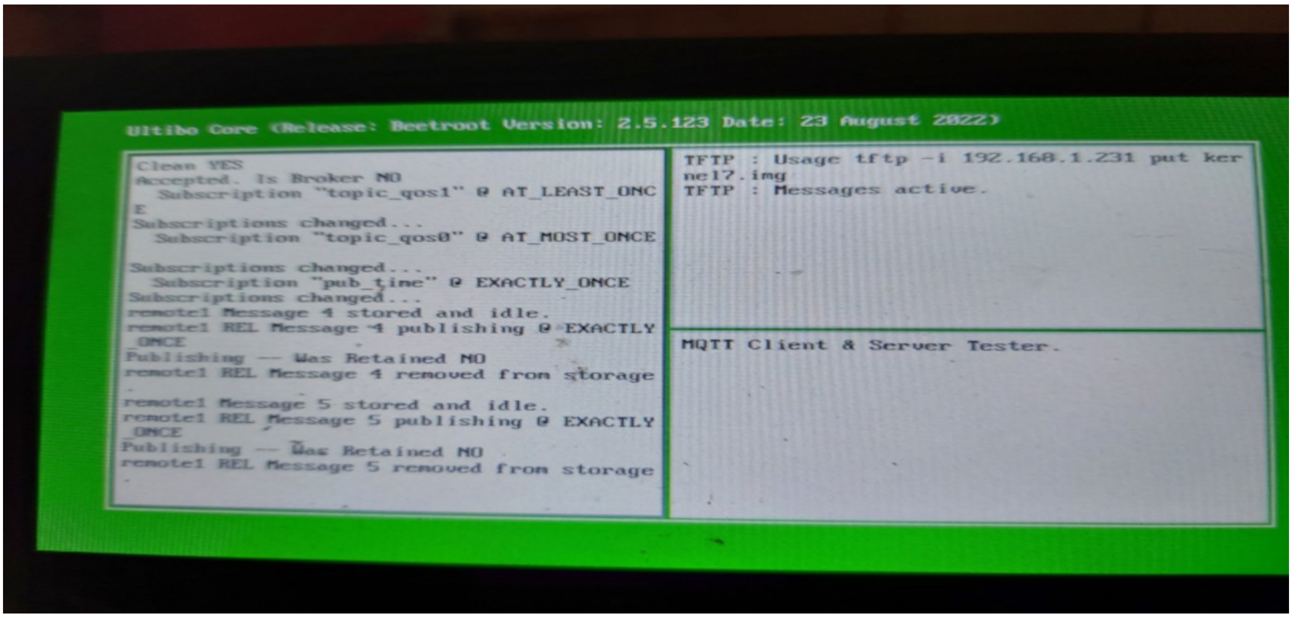
```
Ultibo Core (Release: Beetroot Version: 2.5.123 Date: 23 August 2022)

timer 3 triggered
Thread Created
Connect
  Check User testuser pass password123
Clean YES
Accepted. Is Broker NO
  Subscription "topic_qos1" @ AT_LEAST_ONCE
Subscriptions changed...
  Subscription "topic_qos0" @ AT_MOST_ONCE
Subscriptions changed...
  Subscription "pub_time" @ EXACTLY_ONCE
Subscriptions changed...
remote1 Message 4 stored and idle.
remote1 REL Message 4 publishing @ EXACTLY_ONCE
Publishing -- Was Retained NO
remote1 REL Message 4 removed from storage.

TFTP : Usage tftp -i 10.0.2.15 put kernel7.img
TFTP : Messages active.

MQTT Client & Server Tester.
```

This is a Ultibo RPi3B with 7in display.



```
Ultibo Core (Release: Beetroot Version: 2.5.123 Date: 23 August 2022)

Clean YES
Accepted. Is Broker NO
  Subscription "topic_qos1" @ AT_LEAST_ONC
E
Subscriptions changed...
  Subscription "topic_qos0" @ AT_MOST_ONCE

Subscriptions changed...
  Subscription "pub_time" @ EXACTLY_ONCE
Subscriptions changed...
remote1 Message 4 stored and idle.
remote1 REL Message 4 publishing @ EXACTLY
_ONCE
Publishing -- Was Retained NO
remote1 REL Message 4 removed from storage

remote1 Message 5 stored and idle.
remote1 REL Message 5 publishing @ EXACTLY
_ONCE
Publishing -- Was Retained NO
remote1 REL Message 5 removed from storage

TFTP : Usage tftp -i 192.168.1.231 put ker
nel7.img
TFTP : Messages active.

MQTT Client & Server Tester.
```

https://en.wikipedia.org/wiki/MQTT

MQTT (originally an initialism of **MQ Telemetry Transport**[a]) is a lightweight, publish-subscribe, machine to machine network protocol for message queue/message queuing service. It is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth, such as in the Internet of Things (IoT). It must run over a transport protocol that provides ordered, lossless, bi-directional connections—typically, TCP/IP.[1] It is an open OASIS standard and an ISO recommendation (**ISO/IEC 20922**).

RTC time setting
In the process of converting my "https://github.com/develone/pico_w-remotes.git".
This version used ntp for setting the RTC in Pico_W. The new version "https://github.com/develone/pico_w-mqtt.git" uses a RPI to publish date and time information to topic 'pub_time' and the Pico_W subscribes to topic 'pub_time'.

```
../pub-time pi4-50
2023-04-07-05-38-18
```

In the function "mqtt_incoming_data_cb" parses the received time information and sets the Pico_W RTC.

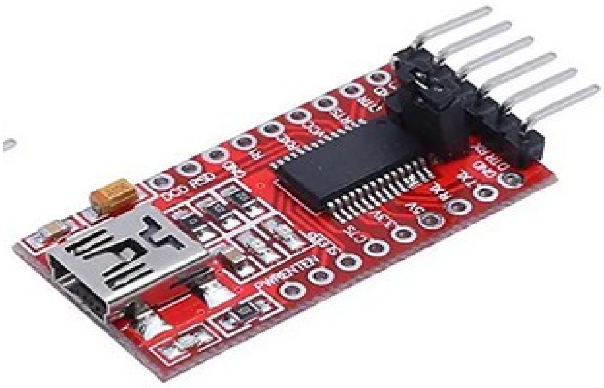
```
t 0x0 &t 0x0 *pt 0x200220a0
t_ntp 0x0 &pt_ntp 0x0 *pt_ntp 0x200220dc
2023
04
07
05
38
18
2023-04-07-05-38-18
2023/04/07 05:38:27
```

Time information is reported to users using tcp_debug socket.

```
../pi_tcp_tests/cli1
Socket created successfully
Connected with server successfully
Starting FreeRTOS on core 0: ver 0.0.02 remote1
Connecting to Wi-Fi...
Connected. iperf server 192.168.1.176 4001
starting watchdog timer task
mqtt_ip = 0xd401a8c0 mqtt_port = 1883
mqtt_connect 0x0 mqtt_connect 0x1
2023-04-07-05-38-18
2023/04/07 05:38:2nnect 0x1
2023/04/07 05:42:37
mqtt_connect 0x1 mqtt_connect 0x1
40:57
mqtt_connect 0x1 mqtt_connect 0x1
2023/04/07 05:41:22
mqtt ���
```

I have several pico_w connected to my home Wifi. Currently a 512 byte debug is sent to RPi4-4GB using (cli1, cli2, cli3, cli4, cli5, and cli6).

GPIO



The USB to UART is currently used to see the debug from pico_w. This will be removed and debug will be available using programs (cli1, cli2, cli3, cli4, cli5, and cli6).

and connected to the RPi4B 4Gb USB to see the debug output.

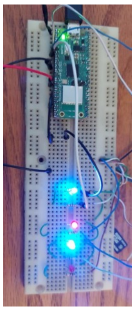
Now this can be done with the programs (cli1, cli2, cli3, cli4, cli5, and cli6).

Examples of the programming & debug are found "https://github.com/develone/pico_w-mqtt/blob/dev/doc/info.txt".
Modified output "https://github.com/develone/pico_w-mqtt/blob/dev/doc/info_1.txt".
The buffer now is 512 bytes. The first 256 is used for booting information and the next 256 are used following the connection to WiFi. Note: mqtt_connected 0 then mqtt_connected 1 which is when the connection to the Mosquitto Broker.

```
 devel@pi4-30:~/pico_w-mqtt/remote5 $ ../pi_tcp_tests/cli1
Socket created successfully
Connected with server successfully
Starting FreeRTOS on core 0: ver 0.0.02 remote1
Connecting to Wi-Fi...
Connected. iperf server 192.168.1.176 4001
starting watchdog timer task
mqtt_ip = 0xd401a8c0 mqtt_port = 1883
mqtt_connect 0x0 mqtt_connect 0x1
2023-04-07-05-38-18
2023/04/07 05:38:2nnect 0x1
2023/04/07 05:42:37
mqtt_connect 0x1 mqtt_connect 0x1
40:57
mqtt_connect 0x1 mqtt_connect 0x1
2023/04/07 05:41:22
mqtt ���
```

GPIO

0	3f	0011 1111
1	06	0000 0110
2	5b	0101 1011
3	4f	0100 1111
4	66	0110 0110
5	6d	0110 1101
6	7d	0111 1101
7	07	0000 0111
8	7f	0111 1111
9	67	0110 0111



```
int bits[10] = {
        0x3f,  // 0
        0x06,  // 1
        0x5b,  // 2
        0x4f,  // 3
        0x66,  // 4
        0x6d,  // 5
        0x7d,  // 6
        0x07,  // 7
        0x7f,  // 8
        0x67   // 9
};
GPIO2 pin 4 pico_w  blue A
GPIO3 pin 5 pico_w  red  B
GPIO4 pin 6 pico_w  blue C
--A--
F   B
--G--
E   C
--D--

void gpio_task(__unused void *params) {
    //bool on = false;
    printf("gpio_task starts\n");



//We could use gpio_set_dir_out_masked() here

    for (int gpio = FIRST_GPIO; gpio < FIRST_GPIO + 7; gpio++) {
        gpio_init(gpio);
        gpio_set_dir(gpio, GPIO_OUT);
        // Our bitmap above has a bit set where we need an LED on, BUT, we are pulling low to light
        // so invert our output
        gpio_set_outover(gpio, GPIO_OVERRIDE_INVERT);
    }

    gpio_init(BUTTON_GPIO);
    gpio_set_dir(BUTTON_GPIO, GPIO_IN);
    // We are using the button to pull down to 0v when pressed, so ensure that when
    // unpressed, it uses internal pull ups. Otherwise when unpressed, the input will
    // be floating.
    gpio_pull_up(BUTTON_GPIO);

    //int val = 0;
    while (true) {
        int val = 0;
        if (!gpio_get(BUTTON_GPIO)) {
            if (val == 9) {
                val = 0;
            } else {
                val++;
            }
        } else if (val == 0) {
            val = 9;
        } else {
            val--;
        }

        // We are starting with GPIO 2, our bitmap starts at bit 0 so shift to start at 2.
        int32_t mask = bits[val] << FIRST_GPIO;
        gpio_set_mask(mask);
        sleep_ms(250);
        gpio_clr_mask(mask);


        vTaskDelay(200);
    }
}
```