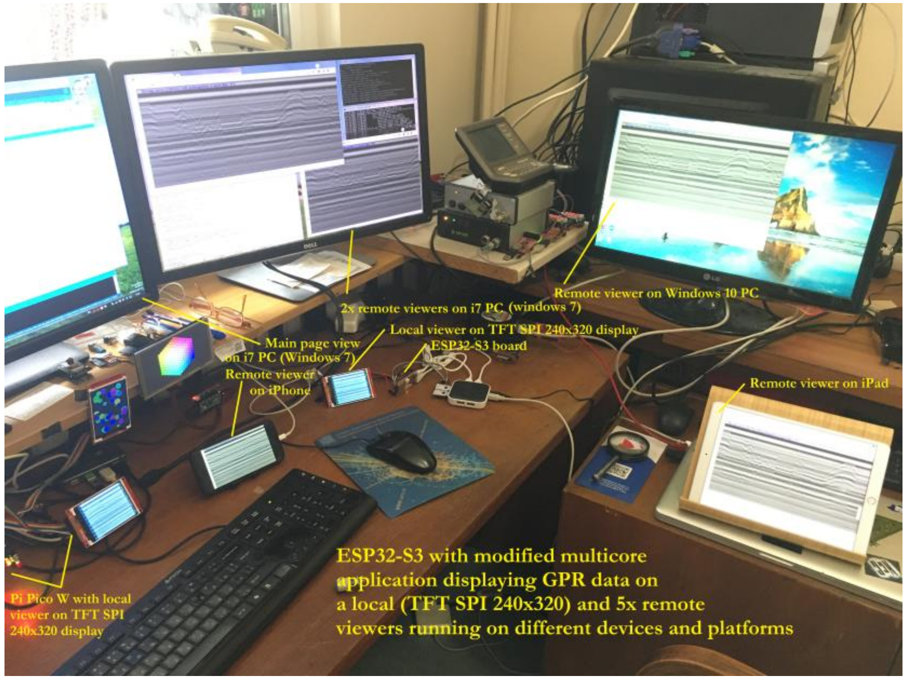
Modified multicore application as the proof-of-concept for local and remote visualization of GPR data via ESP32-S3 board with TFT SPI 240x320 display

The modified multicore application is modification of the unified multicore application for ESP32 and RP2040 with 3.2" TFT SPI 240x320 display to run on RP2040, ESP32-WROOM and ESP32-S3-WROOM based boards. The main bug (impossibility to display and edit SPIFFS files in the file browser in case of compiling application against Espressif ESP32 core 2.0 or later) is fixed by own code instead of using SPIFFSEditor library component (part of the core ESPAsyncWebServer library). SPIFFS structure is changed to separate private and public files. The development process was eased thanks to using USB OTG JTAG/Serial (USBSerial) and UART0 (Serial) interfaces for uploading the program and printing of debug messages respectively. Serial ports on UART1 (Serial1) and UART2 (Serial2) are used to connect GPR and GPS devices.

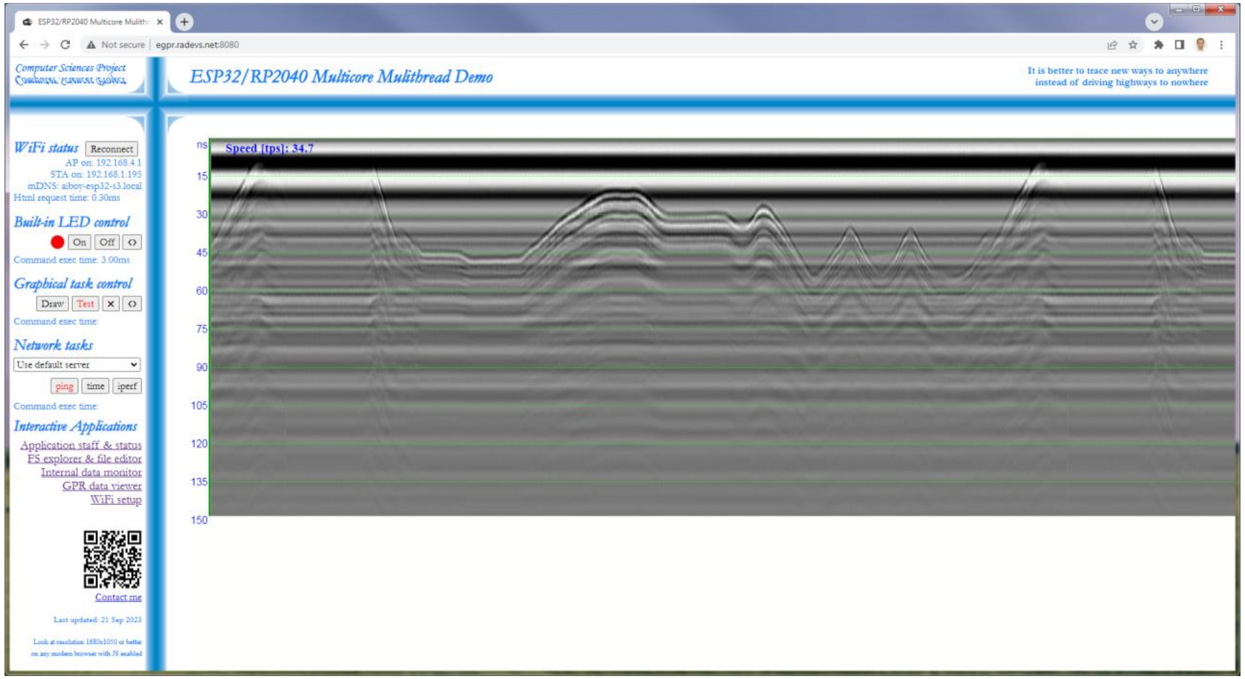
It is also added a code to read GPR data via serial port and display them on local TFT display. A code based on AsyncEventSource is also added to send GPR data to all registered remote clients. Demo version of remote GPR data viewer is developed as web application to prove the concept. As it is visible from the photo below 5x remote clients are served in addition to the local display. There is no disturbance in any of the served local and remote viewers at more than 30 tps (tracks per second). For the moment GPR data are sourced by written in JavaScript simulator reading them from a SEGY file. At the first tests application can run on all Pi Pico W, ESP32-WROOM and ESP32-s3-WROOM based boards but only on ESP32-s3-WROOM one it is working without problems and stable enough.

In the final application for GPR data visualization some of the components in current application will be removed or modified and others will be added. The control of the GPR device by the local and remote viewers is under discussion. In case of remote control of the GPR device the concurrence may cause problems so it has to be assessed its usefulness. There are following alternatives for the local control: touch screen, rotational encoder or buttons. It is cleaner remote viewers to control visualization only. Saving of GPR data to file locally and/or remotely has to be discusses as well.

Modified multicore application in pictures


Remote viewer on Windows 10 PC
2x remote viewers on i7 PC (windows 7)
Local viewer on TFT SPI 240x320 display
ESP32-S3 board
Main page view
on i7 PC (Windows 7)
Remote viewer
on iPhone
Remote viewer on iPad
Pi Pico W with local
viewer on TFT SPI
240x320 display
ESP32-S3 with modified multicore
application displaying GPR data on
a local (TFT SPI 240x320) and 5x remote
viewers running on different devices and platforms

Test suit photo


ESP32/RP2040 Multicore Mulith
Not secure | egpr.radevs.net:8080
Computer Sciences Project
ns
15
30
45
60
75
90
105
120
135
150
Reconnect
On
Off
<>
Draw
Test
×
<>
ping
time
iperf
Contact me

ESP32/RP2040 Multicore Mulithread Demo

It is better to trace new ways to anywhere
instead of driving highways to nowhere

Speed [tps]: 34.7

WiFi status

AP on: 192.168.4.1
STA on: 192.168.1.195
mDNS: aiboy-esp32-s3.local
Html request time: 0.30ms

Built-in LED control

Command exec time: 3.00ms

Graphical task control

Command exec time:

Network tasks

Use default server

Command exec time:

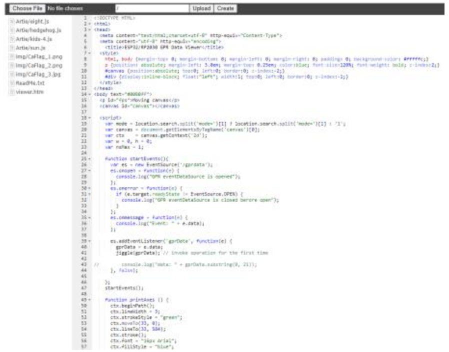
Interactive Applications

Application staff & status
FS explorer & file editor
Internal data monitor
GPR data viewer
WiFi setup

Last updated: 21 Sep 2023

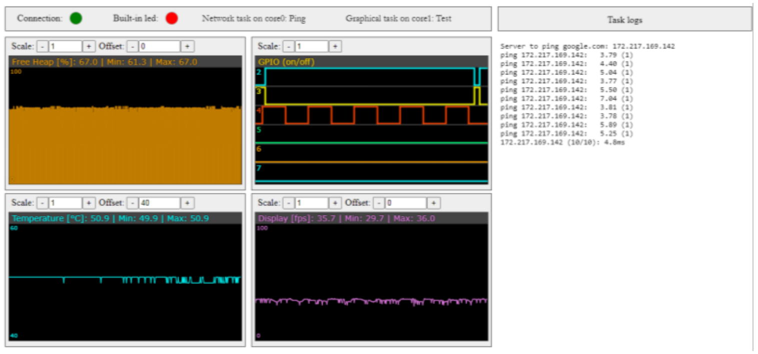
Look at resolution 1680x1050 or better
on any modern browser with JS enabled

Main web page view with remote GPR data viewer



```
<!DOCTYPE HTML>
<html>
<head>
  <meta content="text/html;charset=utf-8" http-equiv="Content-Type">
  <meta content="utf-8" http-equiv="encoding">
    <title>ESP32/RP2040 GPR Data Viewer</title>
  <style>
    html, body {margin-top: 0; margin-bottom: 0; margin-left: 0; margin-right: 0; padding: 0; background-color: #ffffff;}
    p {position: absolute; margin-left: 3.0em; margin-top: 0.25em; color:blue; font-size:120%; font-weight: bold; z-index:2;}
    #canvas {position:absolute; top:0; left:0; border:0; z-index:-2;}
    #div {display:inline-block; float:"left"; width:1; top:0; left:0; border:0; z-index:-1;}
  </style>
</head>
<body text="#0000FF">
  <p id="fps">Moving canvas</p>
  <canvas id="canvas"></canvas>

  <script>
    var mode = location.search.split('mode=')[1] ? location.search.split('mode=')[1] : '1';
    var canvas = document.getElementsByTagName('canvas')[0];
    var ctx    = canvas.getContext('2d');
    var w = 0, h = 0;
    var nsMax = 1;

    function startEvents(){
      var es = new EventSource('/gprdata');
      es.onopen = function(e) {
        console.log("GPR eventDataSource is opened");
      };
      es.onerror = function(e) {
        if (e.target.readyState != EventSource.OPEN) {
          console.log("GPR eventDataSource is closed before open");
        }
      };
      es.onmessage = function(e) {
        console.log("Event: " + e.data);
      };

      es.addEventListener('gprData', function(e) {
        gprData = e.data;
        jiggle(gprData); // invoke operation for the first time

//        console.log("data: " + gprData.substring(0, 21));
      }, false);

    };
    startEvents();

    function printAxes () {
      ctx.beginPath();
      ctx.lineWidth = 3;
      ctx.strokeStyle = "green";
      ctx.moveTo(33, 0);
      ctx.lineTo(33, 504);
      ctx.stroke();
      ctx.font = "16px Arial";
      ctx.fillStyle = "blue";
```

SPIFFS browser


Connection: ●
Built-in led: ●
Scale: - 1 + Offset: - 0 +
100
Scale: - 1 +
2
3
4
5
6
7
Scale: - 1 + Offset: - 40 +
60
40
Scale: - 1 + Offset: - 0 +
100
0
Task logs

Network task on core0: Ping

Graphical task on core1: Test

Free Heap [%]: 67.0 | Min: 61.3 | Max: 67.0

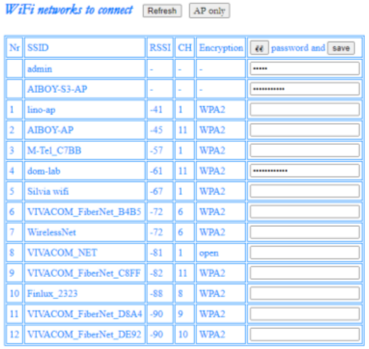
GPIO (on/off)

Temperature [°C]: 50.9 | Min: 49.9 | Max: 50.9

Display [fps]: 35.7 | Min: 29.7 | Max: 36.0

```
Server to ping google.com: 172.217.169.142
ping 172.217.169.142:   3.79 (1)
ping 172.217.169.142:   4.40 (1)
ping 172.217.169.142:   5.04 (1)
ping 172.217.169.142:   3.77 (1)
ping 172.217.169.142:   5.50 (1)
ping 172.217.169.142:   7.04 (1)
ping 172.217.169.142:   3.81 (1)
ping 172.217.169.142:   3.78 (1)
ping 172.217.169.142:   5.89 (1)
ping 172.217.169.142:   5.25 (1)
172.217.169.142 (10/10): 4.8ms
```

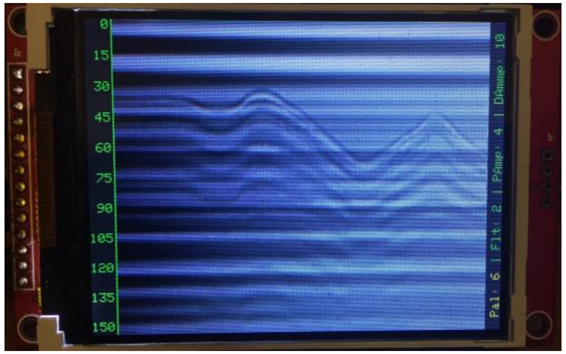
Internal data monitor



WiFi networks to connect

Nr	SSID	RSSI	CH	Encryption	«« password and save
	admin	-	-	-	•••••
	AIBOY-S3-AP	-	-	-	••••••••••••
1	lino-ap	-41	1	WPA2	
2	AIBOY-AP	-45	11	WPA2	
3	M-Tel_C7BB	-57	1	WPA2	
4	dom-lab	-61	11	WPA2	•••••••••••••
5	Silvia wifi	-67	1	WPA2	
6	VIVACOM_FiberNet_B4B5	-72	6	WPA2	
7	WirelessNet	-72	6	WPA2	
8	VIVACOM_NET	-81	1	open	
9	VIVACOM_FiberNet_C8FF	-82	11	WPA2	
10	Finlux_2323	-88	8	WPA2	
11	VIVACOM_FiberNet_D8A4	-90	9	WPA2	
12	VIVACOM_FiberNet_DE92	-90	10	WPA2	

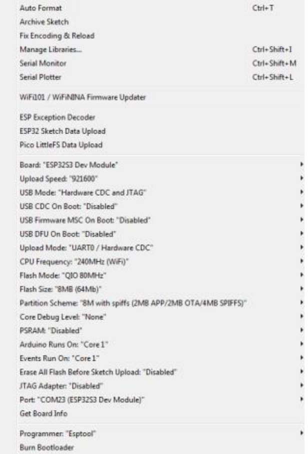
WiFi setup view



TFT SPI display with GPR data

Modified multicore application for ESP32-S3-WROOM-1, RP2040 & CYW43439 and ESP32-WROOM with 3.2” TFT SPI 240x320 display – summary in pictures

ESP32-S3-WROOM-1 (Olimex
ESP32-S3-DevKit-Lipo board)



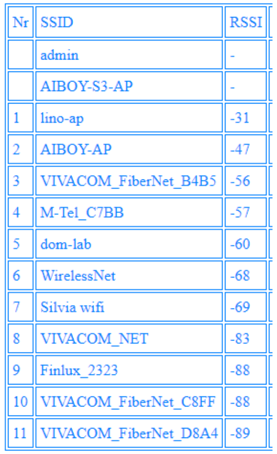
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L

WiFi101 / WiFiNINA Firmware Updater

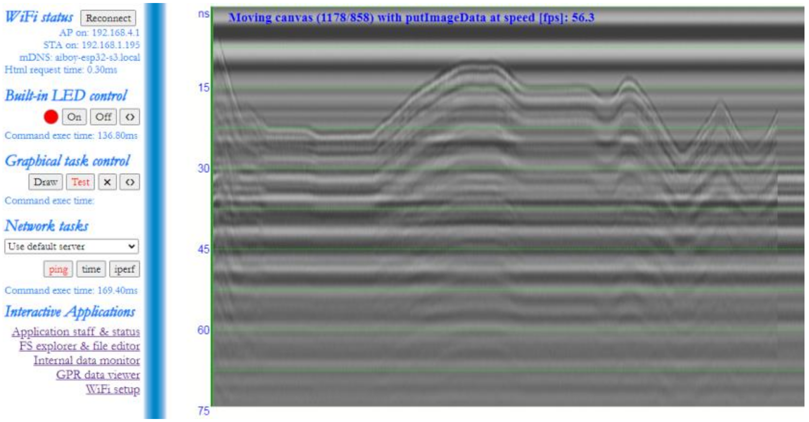
ESP Exception Decoder
ESP32 Sketch Data Upload
Pico LittleFS Data Upload

Board: "ESP32S3 Dev Module"
Upload Speed: "921600"
USB Mode: "Hardware CDC and JTAG"
USB CDC On Boot: "Disabled"
USB Firmware MSC On Boot: "Disabled"
USB DFU On Boot: "Disabled"
Upload Mode: "UART0 / Hardware CDC"
CPU Frequency: "240MHz (WiFi)"
Flash Mode: "QIO 80MHz"
Flash Size: "8MB (64Mb)"
Partition Scheme: "8M with spiffs (2MB APP/2MB OTA/4MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"
Erase All Flash Before Sketch Upload: "Disabled"
JTAG Adapter: "Disabled"
Port: "COM23 (ESP32S3 Dev Module)"
Get Board Info

Programmer: "Esptool"
Burn Bootloader



Nr	SSID	RSSI
	admin	-
	AIBOY-S3-AP	-
1	lino-ap	-31
2	AIBOY-AP	-47
3	VIVACOM_FiberNet_B4B5	-56
4	M-Tel_C7BB	-57
5	dom-lab	-60
6	WirelessNet	-68
7	Silvia wifi	-69
8	VIVACOM_NET	-83
9	Finlux_2323	-88
10	VIVACOM_FiberNet_C8FF	-88
11	VIVACOM_FiberNet_D8A4	-89



WiFi status Reconnect
AP on: 192.168.4.1
STA on: 192.168.1.195
mDNS: aiboy-esp32-s3.local
Html request time: 0.30ms

Built-in LED control
● On Off <>
Command exec time: 136.80ms

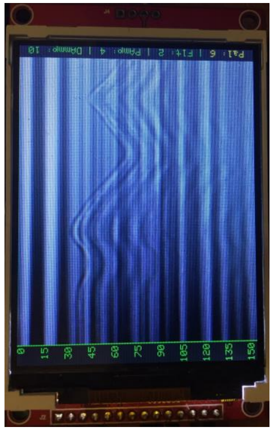
Graphical task control
Draw Test × <>
Command exec time:

Network tasks
Use default server
ping time iperf
Command exec time: 169.40ms

Interactive Applications
Application staff & status
FS explorer & file editor
Internal data monitor
GPR data viewer
WiFi setup

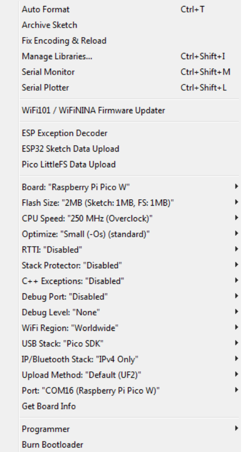
Moving canvas (1178/858) with putImageData at speed [fps]: 56.3

```
Sketch uses 900633 bytes (42%) of program storage space. Maximum is 2097152 bytes.
Global variables use 51152 bytes (15%) of dynamic memory, leaving 276528 bytes for local variables. Maximum is 327680 bytes.
```



56 tps @ 460800 bps

RP2040 & CYW43439
(Raspberry Pi Pico W board)



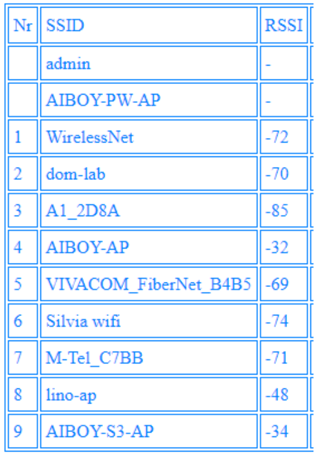
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L

WiFi101 / WiFiNINA Firmware Updater

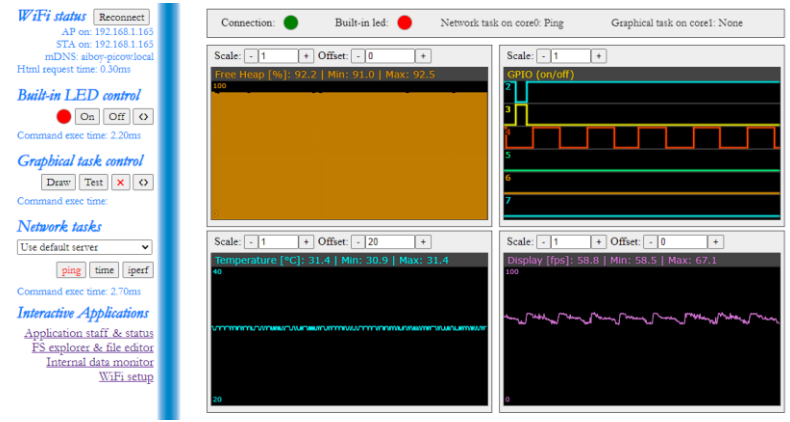
ESP Exception Decoder
ESP32 Sketch Data Upload
Pico LittleFS Data Upload

Board: "Raspberry Pi Pico W"
Flash Size: "2MB (Sketch: 1MB, FS: 1MB)"
CPU Speed: "250 MHz (Overclock)"
Optimize: "Small (-Os) (standard)"
RTTI: "Disabled"
Stack Protector: "Disabled"
C++ Exceptions: "Disabled"
Debug Port: "Disabled"
Debug Level: "None"
WiFi Region: "Worldwide"
USB Stack: "Pico SDK"
IP/Bluetooth Stack: "IPv4 Only"
Upload Method: "Default (UF2)"
Port: "COM16 (Raspberry Pi Pico W)"
Get Board Info

Programmer
Burn Bootloader



Nr	SSID	RSSI
	admin	-
	AIBOY-PW-AP	-
1	WirelessNet	-72
2	dom-lab	-70
3	A1_2D8A	-85
4	AIBOY-AP	-32
5	VIVACOM_FiberNet_B4B5	-69
6	Silvia wifi	-74
7	M-Tel_C7BB	-71
8	lino-ap	-48
9	AIBOY-S3-AP	-34



WiFi status Reconnect
AP on: 192.168.1.165
STA on: 192.168.1.165
mDNS: aiboy-picow.local
Html request time: 0.30ms

Built-in LED control
● On Off <>
Command exec time: 2.20ms

Graphical task control
Draw Test × <>
Command exec time:

Network tasks
Use default server
ping time iperf
Command exec time: 2.70ms

Interactive Applications
Application staff & status
FS explorer & file editor
Internal data monitor
WiFi setup

Connection: ● Built-in led: ● Network task on core0: Ping Graphical task on core1: None

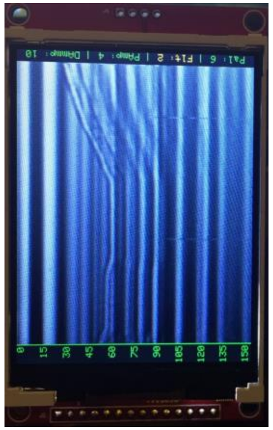
Scale: - 1 + Offset: - 0 +
Free Heap [%]: 92.2 | Min: 91.0 | Max: 92.5

Scale: - 1 +
GPIO (on/off)

Scale: - 1 + Offset: - 20 +
Temperature [°C]: 31.4 | Min: 30.9 | Max: 31.4

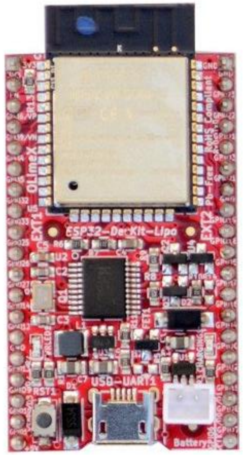
Scale: - 1 + Offset: - 0 +
Display [fps]: 58.8 | Min: 58.5 | Max: 67.1

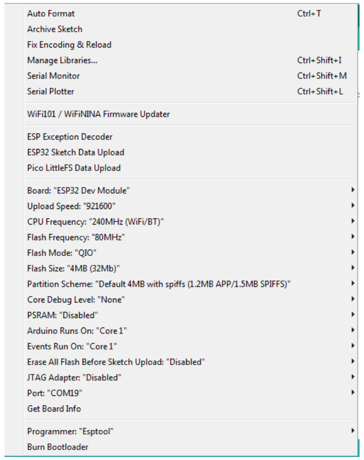
```
Sketch uses 469072 bytes (44%) of program storage space. Maximum is 1044480 bytes.
Global variables use 74688 bytes (28%) of dynamic memory, leaving 187456 bytes for local variables. Maximum is 262144 bytes.
```



67 tps @ 460800 bps

ESP32-WROOM (Olimex
ESP32-DevKit-Lipo board)





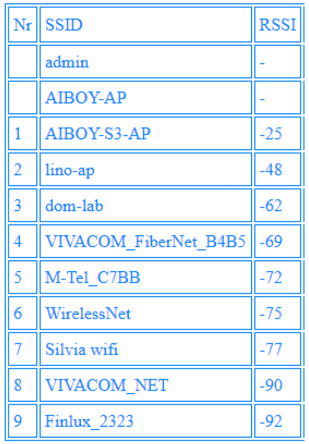
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L

WiFi101 / WiFiNINA Firmware Updater

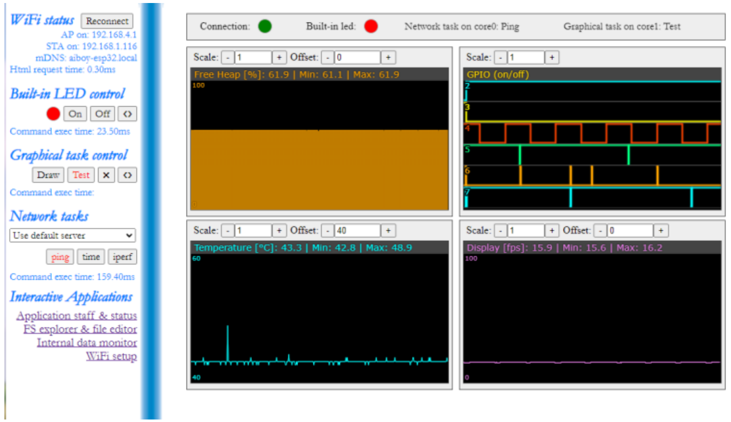
ESP Exception Decoder
ESP32 Sketch Data Upload
Pico LittleFS Data Upload

Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"
Erase All Flash Before Sketch Upload: "Disabled"
JTAG Adapter: "Disabled"
Port: "COM19"
Get Board Info

Programmer: "Esptool"
Burn Bootloader



Nr	SSID	RSSI
	admin	-
	AIBOY-AP	-
1	AIBOY-S3-AP	-25
2	lino-ap	-48
3	dom-lab	-62
4	VIVACOM_FiberNet_B4B5	-69
5	M-Tel_C7BB	-72
6	WirelessNet	-75
7	Silvia wifi	-77
8	VIVACOM_NET	-90
9	Finlux_2323	-92



WiFi status Reconnect
AP on: 192.168.4.1
STA on: 192.168.1.116
mDNS: aiboy-esp32.local
Html request time: 0.30ms

Built-in LED control
● On Off <>
Command exec time: 23.50ms

Graphical task control
Draw Test × <>
Command exec time:

Network tasks
Use default server
ping time iperf
Command exec time: 159.40ms

Interactive Applications
Application staff & status
FS explorer & file editor
Internal data monitor
WiFi setup

Connection: ● Built-in led: ● Network task on core0: Ping Graphical task on core1: Test

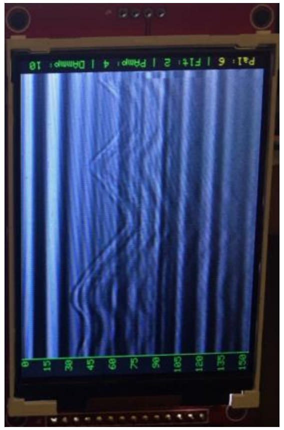
Scale: - 1 + Offset: - 0 +
Free Heap [%]: 61.9 | Min: 61.1 | Max: 61.9

Scale: - 1 +
GPIO (on/off)

Scale: - 1 + Offset: - 40 +
Temperature [°C]: 43.3 | Min: 42.8 | Max: 48.9

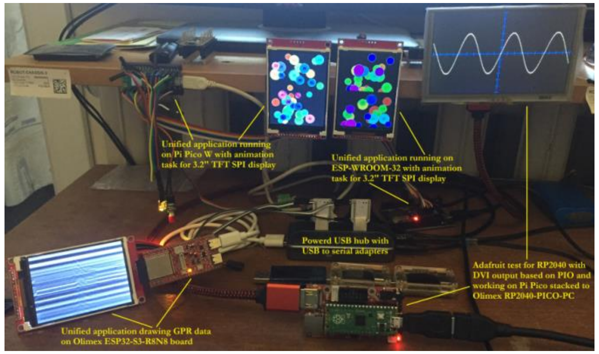
Scale: - 1 + Offset: - 0 +
Display [fps]: 15.9 | Min: 15.6 | Max: 16.2

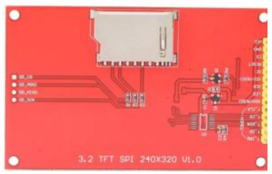
```
Sketch uses 940097 bytes (71%) of program storage space. Maximum is 1310720 bytes.
Global variables use 47904 bytes (14%) of dynamic memory, leaving 279776 bytes for local variables. Maximum is 327680 bytes.
```

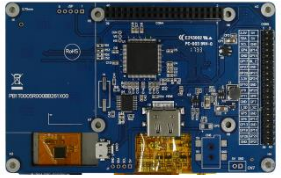


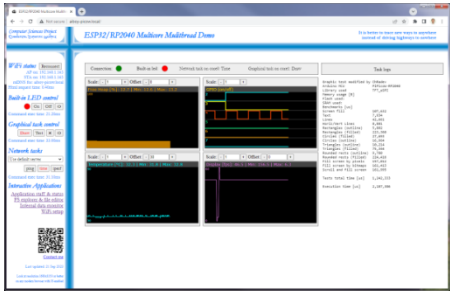
16 tps @ 115200 bps

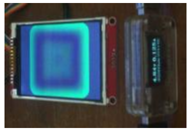
Unified multicore application for ESP32, RP2040 with 3.2” TFT SPI 240x320 display – summary in pictures and projects history

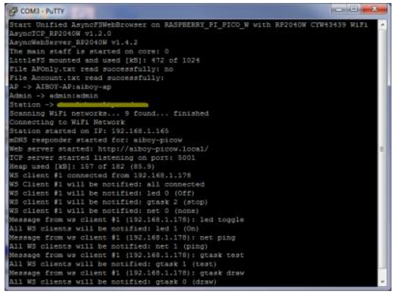

Unified application running on Pi Pico W with animation task for 3.2" TFT SPI display
Unified application running on ESP-WROOM-32 with animation task for 3.2" TFT SPI display
Powerd USB hub with USB to serial adapters
Adafruit test for RP2040 with DVI output based on PIO and working on Pi Pico stacked to Olimex RP2040-PICO-PC
Unified application drawing GPR data on Olimex ESP32-S3-R8N8 board



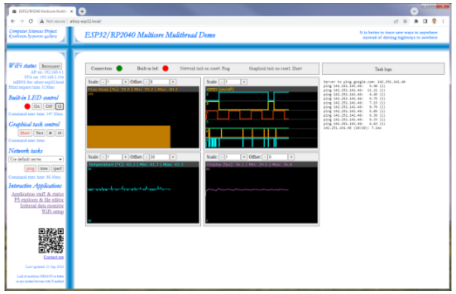


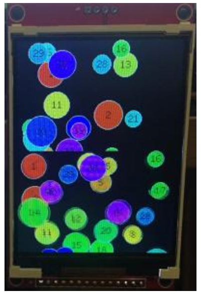


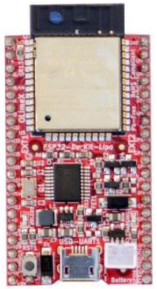


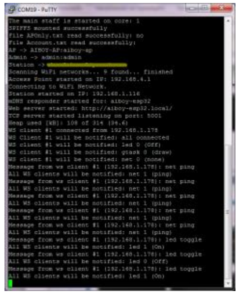


```
Start Unified AsyncFSWebBrowser on RASPBERRY_PI_PICO_W with RP2040W CYW43439 WiFi
AsyncTCP_RP2040W v1.2.0
AsyncWebServer_RP2040W v1.4.2
The main staff is started on core: 0
LittleFS mounted and used [kB]: 472 of 1024
File APOnly.txt read successfully: no
File Account.txt read successfully:
AP -> AIBOY-AP:aiboy-ap
Admin -> admin:admin
Station -> [redacted]
Scanning WiFi networks... 9 found... finished
Connecting to WiFi Network
Station started on IP: 192.168.1.165
mDNS responder started for: aiboy-picow
Web server started: http://aiboy-picow.local/
TCP server started listening on port: 5001
Heap used [kB]: 157 of 182 (85.9)
WS client #1 connected from 192.168.1.178
WS Client #1 will be notified: all connected
WS client #1 will be notified: led 0 (Off)
WS client #1 will be notified: gtask 2 (stop)
WS client #1 will be notified: net 0 (none)
Message from ws client #1 (192.168.1.178): led toggle
All WS clients will be notified: led 1 (On)
Message from ws client #1 (192.168.1.178): net ping
All WS clients will be notified: net 1 (ping)
Message from ws client #1 (192.168.1.178): gtask test
All WS clients will be notified: gtask 1 (test)
Message from ws client #1 (192.168.1.178): gtask draw
All WS clients will be notified: gtask 0 (draw)
```

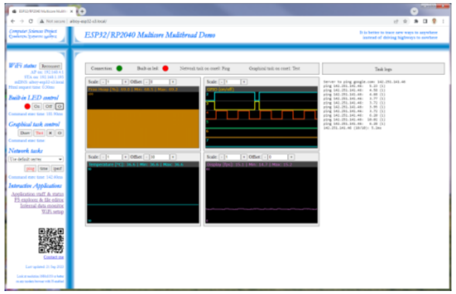


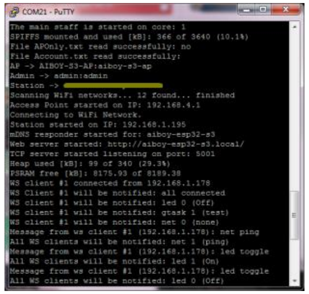






```
The main staff is started on core: 1
SPIFFS mounted successfully
File APOnly.txt read successfully: no
File Account.txt read successfully:
AP -> AIBOY-AP:aiboy-ap
Admin -> admin:admin
Station -> [redacted]
Scanning WiFi networks... 9 found... finished
Access Point started on IP: 192.168.4.1
Connecting to WiFi Network.
Station started on IP: 192.168.1.116
mDNS responder started for: aiboy-esp32
Web server started: http://aiboy-esp32.local/
TCP server started listening on port: 5001
Heap used [kB]: 108 of 314 (34.6)
WS client #1 connected from 192.168.1.178
WS Client #1 will be notified: all connected
WS client #1 will be notified: led 0 (Off)
WS client #1 will be notified: gtask 0 (draw)
WS client #1 will be notified: net 0 (none)
Message from ws client #1 (192.168.1.178): net ping
All WS clients will be notified: net 1 (ping)
Message from ws client #1 (192.168.1.178): net ping
All WS clients will be notified: net 1 (ping)
Message from ws client #1 (192.168.1.178): net ping
All WS clients will be notified: net 1 (ping)
Message from ws client #1 (192.168.1.178): net ping
All WS clients will be notified: net 1 (ping)
Message from ws client #1 (192.168.1.178): net ping
All WS clients will be notified: net 1 (ping)
Message from ws client #1 (192.168.1.178): led toggle
All WS clients will be notified: led 1 (On)
Message from ws client #1 (192.168.1.178): led toggle
All WS clients will be notified: led 0 (Off)
Message from ws client #1 (192.168.1.178): led toggle
All WS clients will be notified: led 1 (On)
```





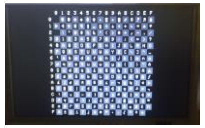
```
The main staff is started on core: 1
SPIFFS mounted and used [kB]: 366 of 3640 (10.1%)
File APOnly.txt read successfully: no
File Account.txt read successfully:
AP -> AIBOY-S3-AP:aiboy-s3-ap
Admin -> admin:admin
Station -> [redacted]
Scanning WiFi networks... 12 found... finished
Access Point started on IP: 192.168.4.1
Connecting to WiFi Network.
Station started on IP: 192.168.1.195
mDNS responder started for: aiboy-esp32-s3
Web server started: http://aiboy-esp32-s3.local/
TCP server started listening on port: 5001
Heap used [kB]: 99 of 340 (29.3%)
PSRAM free [kB]: 8175.93 of 8189.38
WS client #1 connected from 192.168.1.178
WS Client #1 will be notified: all connected
WS client #1 will be notified: led 0 (Off)
WS client #1 will be notified: gtask 1 (test)
WS client #1 will be notified: net 0 (none)
Message from ws client #1 (192.168.1.178): net ping
All WS clients will be notified: net 1 (ping)
Message from ws client #1 (192.168.1.178): led toggle
All WS clients will be notified: led 1 (On)
Message from ws client #1 (192.168.1.178): led toggle
All WS clients will be notified: led 0 (Off)
```

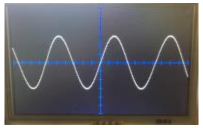
Projects history

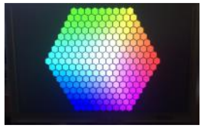
- First cycle of tests included unified graphics test running on Arduino UNO (ATMega328), Arduino Leonardo (ATMega32u4), Arduino D1 R32 / ESP32, Raspberry Pi Pico W (RP2040) and Self-made AVR128db48 boards connected to 3.2” TFT SPI 240x320 display. Application is based on Adafruit performance tests for Adaruit_ILI9341 / Adafruit_GFX and adapted for TFT_ILI9341 and TFT_eSPI libraries. Meanwhile Olimex ESP32-S2 (WROOM and WROVER) boards were tested with multitasking “Hello world & RGB LED”. Another test done is based on ESP32-CAM module as a base of own implementation of Wifi Camera Robot Car project. As a result following open source projects are posted on GitHub:
  - Unified-ILI9341-Graphic-Test
  - Unified-ILI9341-Graphic-Test-plus
- Next cycle of tests was performance assessment of networking capabilities of WiFi equipped ESP32 and Pi Pico W boards. Test applications are based on Arduino libraries ESPAsyncWebServer and AsyncWebServer_RP2040W. Special attention was paid to asynchronous web services, web sockets, WiFi management and unification possibility for both ESP32 and Pi Pico W platforms.
- Next step done was to adapt DrawWithDMA sketch created by Bodmer as example for TFT_eSPI library to work on ESP-WROOM-32 and RP2040 boards. Modified sketch is posted on GitHub as open source:
  - DrawWithDMA
- Next cycle of tests was directed to multicore task execution on dual core versions of ESP32 and RP2040 based boards. United version of AsyncFSBrowser demo with Unified graphic test (TFT-eSPI library case) and modified DrawWithDMA sketch was implemented as multi-file Arduino IDE project running on both Arduino D1 R32 ESP32 (ESP32-WROOM) and Raspberry Pi Pico W (RP2040) boards. It includes web server with web sockets service, TCP server for network performance assessment, internal SPI Flash FS file viewer and editor, monitor showing graphs of the free heap memory, the GPIO states, the internal temperature and the animation frame rate. It also includes accounts management of WiFi in AP and/or STA modes. Graphics part of the application is implemented as tasks alternatively running on second CPU core. Control is based on web sockets and includes built-in LED, running of network tasks like ping, time, iperf and switching of graphic tasks (Adafruit tests and DrawWithDMA animation). Results of network commands and Adafruit tests are printed on monitoring web page. Unified multicore application will be posted on GitHub as soon as become more stable.
- Next cycle of tests started is modification of Unified multicore application for working on ESP32-S3-R8N8 (Olimex ESP32-S3-DevKit-Lipo) to display locally and remotely GPR (Ground Penetrating Radar) data currently generated by simulator. First test done shows that displaying data locally on 3.2” TFT SPI 240x320 display is stable at speeds 60+ tracks per second (at 460800 bps over serial) while all network services work on the second CPU core.
- Next cycle of tests started is experimenting with Pi Pico PIO engine functionality. It was used Raspberry Pi Pico, Olimex RP2040-PICO-PC boards and 5” TFT HDMI 800x480 display and as a beginning adapted by Adafruit Arduino IDE version of PicoDVI library and example tests were running successfully.

All the time the performance table (next page) was updated with the benchmark results measured by Adafruit graphics and DrawWithDMA tests. The connection table (page 5) was also updated.

Full color PicoDVI test on Raspberry Pi Pico, RP2040-PICO-PC and 5” TFT HDMI 800x480 display







Benchmark of unified graphic and scroll tests built on Adafruit_ILI9341, TFT_ILI9341 and TFT_eSPI libraries

Arduino board / MCU	UNO / ATMega328			Leonardo / ATMega32u4			D1 R32 / ESP32			Pi Pico / RP2040			Pi Pico / RP2040 (Overclocked)			Unified App	AVR128db48	ESP32-S3
ILI9341 Library used (SPI clock)	Adafruit	TFT	Speed up	Adafruit	TFT	Speed up	Adafruit (3MHz)	TFT_eSPI	Speed up	Adafruit	TFT_eSPI (27MHz)	Speed up	Adafruit	TFT_eSPI (62.5MHz)	Speed up	TFT_eSPI with DMA	Adafruit	Adafruit (27MHz)
Memory usage [B]																		
Flash used:	23,736 of 32,256 (73.59%)	21,870 of 32,256 (67.80%)		25,874 of 28,672 (90.24%)	23,992 of 28,672 (83.68%)		237,600 of 1,310,720 (18.13%)	295,261 of 1,310,720 (22.52%)		327,772 of 2,093,056 (15.65%)	372,092 of 2,093,056 (17.78%)		327,868 of 1,568,768 (20%)	372,180 of 1,568,768 (23%)		505,232 of 1,044,480 (48%)	24,354 of 130,560 (18.65%)	295,261 of 1,310,720 (22.52%)
SRAM used:	950 of 2,048 (46.39%)	746 of 2,048 (36.43%)		915 of 2,560 (35.74%)	711 of 2,560 (27.77%)		37,264 of 327,680 (11.37%)	19,480 of 327,680 (5.94%)		71,324 of 262,144 (27.21%)	71,768 of 262,144 (27.38%)		71,324 of 262,144 (27%)	71,768 of 262,144 (27%)		74,912 of 262,144 (28%)	1,087 of 16,384 (6.63%)	19,480 of 327,680 (5.94%)
Benchmarks [us]								~42°C			~32°C			~34°C		~36°C		~50°C
Screen fill	1,496,456	870,220	1.720	1,503,900	874,600	1.720	2,120,993	274,575	9.097	604,056	281,577	2.145	497,451	107,972	4.607	107,567	1,603,604	274,575
Text	147,088	60,416	2.435	147,820	60,724	2.434	99,610	32,599	6.491	45,452	18,831	2.414	30,599	8,085	3.785	8,070	114,885	32,599
Lines	1,172,116	242,732	4.829	1,178,004	243,988	4.828	986,748	339,491	10.975	454,856	101,897	4.464	304,234	42,741	7.118	43,648	946,199	339,491
Horiz/Vert Lines	125,064	71,336	1.753	125,656	71,696	1.753	173,171	24,171	8.603	50,042	23,541	2.126	40,853	9,078	4.500	8,880	132,637	24,171
Rectangles (outline)	82,228	45,844	1.794	82,632	46,076	1.793	110,682	15,996	8.697	32,657	14,932	2.187	26,417	5,773	4.576	5,686	85,703	15,996
Rectangles (filled)	3,107,060	1,807,436	1.719	3,122,844	1,816,740	1.719	4,402,687	570,510	9.096	1,253,856	584,372	2.146	1,032,576	224,086	4.608	223,506	3,329,307	570,510
Circles (filled)	452,728	284,064	1.594	454,916	285,536	1.593	492,735	95,809	7.704	167,914	71,149	2.360	126,969	28,025	4.531	27,896	423,221	95,809
Circles (outline)	497,252	135,580	3.668	499,604	136,148	3.670	432,728	150,143	12.978	199,626	37,258	5.358	133,263	15,561	8.564	15,743	404,412	150,143
Triangles (outline)	261,056	59,496	4.388	262,392	59,808	4.387	225,959	74,819	10.265	101,400	23,636	4.290	68,473	10,319	6.636	10,463	213,681	74,819
Triangles (filled)	1,330,720	694,456	1.916	1,337,200	698,032	1.916	1,432,757	209,558	8.691	429,998	195,995	2.194	345,244	75,450	4.576	75,102	1,279,412	209,558
Rounded rects (outline)	228,892	100,004	2.289	230,024	100,532	2.288	230,767	62,675	11.013	92,280	23,635	3.904	65,233	9,576	6.812	9,602	200,582	62,675
Rounded rects (filled)	3,127,968	1,976,936	1.582	3,143,588	1,987,180	1.582	4,384,111	578,880	8.995	1,257,871	586,292	2.145	1,032,024	225,027	4.586	224,252	3,330,751	578,880
Fill screen by pixels	3,369,992	918,732	3.668	3,387,308	923,492	3.668	2,783,609	1,591,181	3.331	1,255,234	504,753	2.487	805,373	229,258	3.513	159,327	2,964,859	1,591,181
Fill screen by bitmaps	528,576	855,088	0.618	531,112	859,520	0.618	435,203	62,752	0.518	66,438	520,180	0.128	70,363	234,904	0.300	166,092	453,099	62,752
Scroll and fill screen	**532,988**	**855,696**	**0.623**	**535,808**	**860,132**	**0.623**	**439,860**	**67,668**	**0.520**	**69,357**	**521,011**	**0.133**	**71,933**	**235,385**	**0.306**	**166,606**	**457,946**	**67,668**
Min	82,228	45,844		82,632	46,076		99,610	15,996		32,657	14,932		26,417	5,773		5,686	85,703	15,996
Avg	**1,097,346**	**598,536**	**1.833**	**1,102,854**	**601,614**	**1.833**	**1,250,108**	**276,722**	**4.497**	**405,402**	**233,937**	**1.733**	**310,067**	**97,416**	**3.183**	**83,496**	**1,062,687**	**276,722**
Max	3,369,992	1,976,936		3,387,308	1,987,180		4,402,687	1,591,181		1,257,871	586,292		1,032,576	235,385		224,252	3,330,751	1,591,181
Sum	16,460,184	8,978,036		16,542,808	9,024,204		18,751,620	4,150,827		6,081,037	3,509,059		4,651,005	1,461,240		1,252,440	15,940,298	4,150,827
DrawWithDMA test (bounsing of 42 colored and numbered circles)								**36fps (Unified App)**			**17.8 fps at CPU 133MHz SPI 27MHz**			**46.5 fps at CPU 250MHz SPI 62.5MHz**	**2.6**	**46.5 fps at CPU 250MHz SPI 62.5MHz**		**60+ tps at GPR data visualisation**

Notes:

- Memory usage numbers are as reported in runtime and slightly different than one reported by the compiler;
- Preparing of the data for filling the screen by pixels or bitmaps are made to be as fast as possible;
- Numbers for “Scroll and fill screen” tests at TFT_ILI9341 and TFT_eSPI libraries should be revised;
- At combination ESP32 and Adafruit_ILI9341 library SPI frequency was lowered to 3MHz while in case of ESP32 S3 SPI frequency can be increased up to 27MHz but in unified application with WiFi networking TFT_eSPI library has some problems especially at using DMA;
- Numbers in “Speed up” column means the operation is that many times faster;
- Overclocking in case of Pi Pico includes increasing of SPI and CPU speeds up to 62.5MHz and 250MHz respectively and application of suggested solution by Bodemar in his Github issue 1460 (working reliably even with 30cm long wires);
- Cases with ESP32 (Unified App), overclocked RP2040 (Unified App) and ESP32-S3 (colored in light violet) were measured by Unified multicore application (in combination with AsyncFSWebBrowser).

Useful links for display of animation with DMA and speed assessment:

Raspberry Pi Pico with ILI9341 TFT and TFT_eSPI Arduino library using RAM & DMA
https://forum.arduino.cc/t/tft_espi-support-for-raspberry-pi-pico-added/702551
https://www.youtube.com/watch?v=njFXIzCTQ_Q
https://github.com/Bodmer/TFT_eSPI/issues/1460#issuecomment-1006661452

This application uses two sprites in RAM and DMA for filling display half buffer while updating the other half. The ILI9341 display operates reliably on Pi Pico up to 62.5MHz so frame rate up to ~43fps is possible with DMA. Overclocking CPU to 250MHz and applying Bodmer note makes it possible frame rates to go up to 46.5fps. The total consumption in overclocked mode of both Pi Pico and SPI TFT is 110mA. The application is unified to run on both RP2040 and ESP-WROOM-32 boards. In case of ESP32 frame rate was lower (~36fps).

Modified multicore application for ESP32-S3, RP2040 and ESP32 to display GPR data locally and remotely

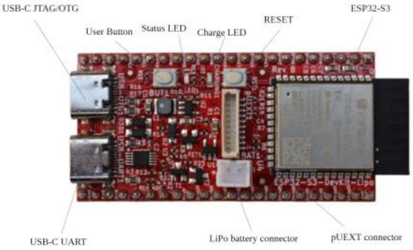
Modified multicore application is based on Unified multicore application which is combination of AsyncFSBrowser demo application, Unified graphic test (TFT-eSPI library) and DrawWithDMA sketches. It is targeted to ESP32-S3 based Olimex ESP32-S3-DevKit-Lipo board.

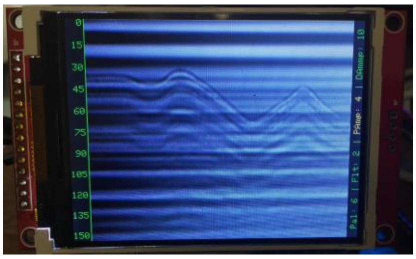
ESP32-S3 is a dual-core XTensa LX7 MCU, capable of running at 240 MHz. Apart from its 512 KB of internal SRAM, it also comes with integrated 2.4 GHz, 802.11 b/g/n Wi-Fi and Bluetooth 5 (LE) connectivity that provides long-range support. It has 45 programmable GPIOs and supports a rich set of peripherals. ESP32-S3 supports larger, high-speed octal SPI flash, and PSRAM with configurable data and instruction cache.

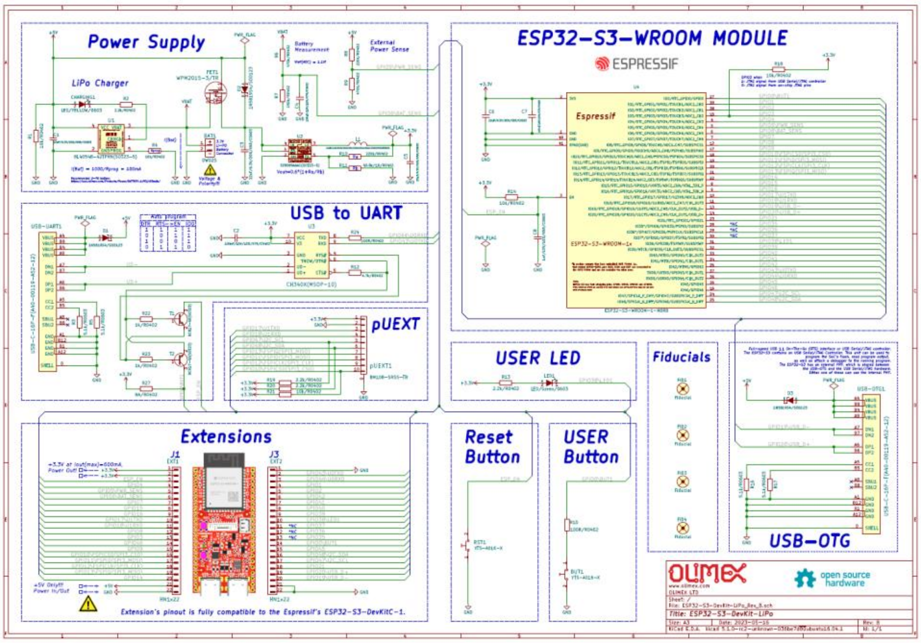
Olimex ESP32-S3-DevKit-Lipo board with ESP32-S3-WROOM-1-N8R8 has 8MB PSRAM and 8MB SPI Flash. It also has pUEXT and 2x USB C connectors (via CH340 USB-serial adapter and native OTG JTAG/Serial with on-chip PHY), LiPo battery charger and connector, user and reset buttons and user and charge LEDs. All GPIO pins are routed to 2x22 pins connectors compatible to the Espressif ESP32-S3-DevJitC-1.

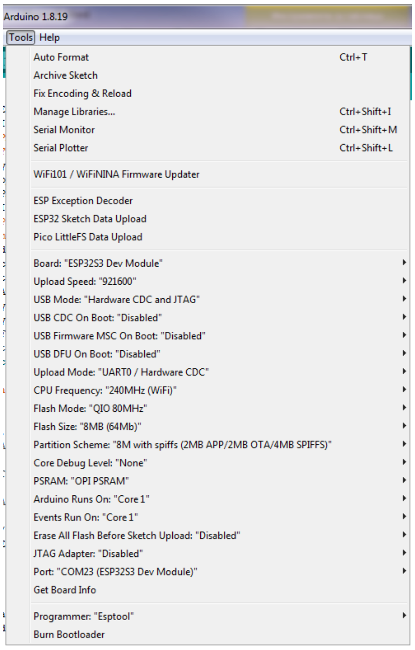
Development is made on Windows 7 / 10 using Arduino IDE (ver. 1.8.9), the latest Espressif system version 2.0.14, AsyncWebServer, Adafruit_ILI9341, Adafruit_GFX etc. libraries (all latest versions). Node.JS (ver. 12.22.9) with serialport library (ver. 8.0.5) is used to develop and use “gpr-simulator” application.

To power and connect the board to the computer may use powered 4+ USB Hub, 2x USB A to USB C cables and 2x CP2102 USB-to-Serial adapters. Install CH340 and CP210x drivers for Windows if required.









Arduino 1.8.19

Tools Help

Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L

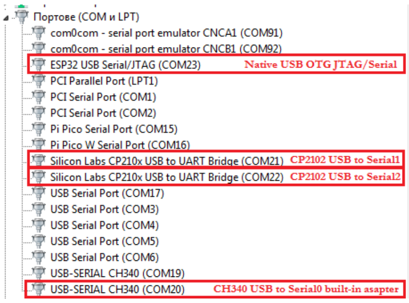
WiFi101 / WiFiNINA Firmware Updater

ESP Exception Decoder
ESP32 Sketch Data Upload
Pico LittleFS Data Upload

Board: "ESP32S3 Dev Module"
Upload Speed: "921600"
USB Mode: "Hardware CDC and JTAG"
USB CDC On Boot: "Disabled"
USB Firmware MSC On Boot: "Disabled"
USB DFU On Boot: "Disabled"
Upload Mode: "UART0 / Hardware CDC"
CPU Frequency: "240MHz (WiFi)"
Flash Mode: "QIO 80MHz"
Flash Size: "8MB (64Mb)"
Partition Scheme: "8M with spiffs (2MB APP/2MB OTA/4MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "OPI PSRAM"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"
Erase All Flash Before Sketch Upload: "Disabled"
JTAG Adapter: "Disabled"
Port: "COM23 (ESP32S3 Dev Module)"
Get Board Info

Programmer: "Esptool"
Burn Bootloader

```
For user configuration of internal SPI flash do following changes:
In ESP32 boards file: C:\Users\BI\AppData\Local\Arduino15\packages\esp32\hardware\esp32\2.0.14\boards.txt
add following lines to esp32s3.name=ESP32S3 Dev Module section:
    esp32s3.menu.PartitionScheme.users_8MB=8M with spiffs (2MB APP/2MB OTA/4MB SPIFFS)
    esp32s3.menu.PartitionScheme.users_8MB.build.partitions=users_8MB
    esp32s3.menu.PartitionScheme.users_8MB.upload.maximum_size=2097152
Add: C:\Users\BI\AppData\Local\Arduino15\packages\esp32\hardware\esp32\2.0.14\tools\partitions\users_8MB.csv
with following content:
    # Name,   Type, SubType, Offset,  Size, Flags
    nvs,      data, nvs,     0x9000,  0x5000,
    otadata,  data, ota,     0xe000,  0x2000,
    app0,     app,  ota_0,   0x10000, 0x200000,
    app1,     app,  ota_1,   0x210000,0x200000,
    spiffs,   data, spiffs,  0x410000,0x3e0000,
    coredump, data, coredump,0x7F0000,0x10000,
Restart Arwuino IDE changes to take effect
```



Портове (COM и LPT)
com0com - serial port emulator CNCA1 (COM91)
com0com - serial port emulator CNCB1 (COM92)
ESP32 USB Serial/JTAG (COM23) Native USB OTG JTAG/Serial
PCI Parallel Port (LPT1)
PCI Serial Port (COM1)
PCI Serial Port (COM2)
Pi Pico Serial Port (COM15)
Pi Pico W Serial Port (COM16)
Silicon Labs CP210x USB to UART Bridge (COM21) CP2102 USB to Serial1
Silicon Labs CP210x USB to UART Bridge (COM22) CP2102 USB to Serial2
USB Serial Port (COM17)
USB Serial Port (COM3)
USB Serial Port (COM4)
USB Serial Port (COM5)
USB Serial Port (COM6)
USB-SERIAL CH340 (COM19)
USB-SERIAL CH340 (COM20) CH340 USB to Serial0 built-in asapter

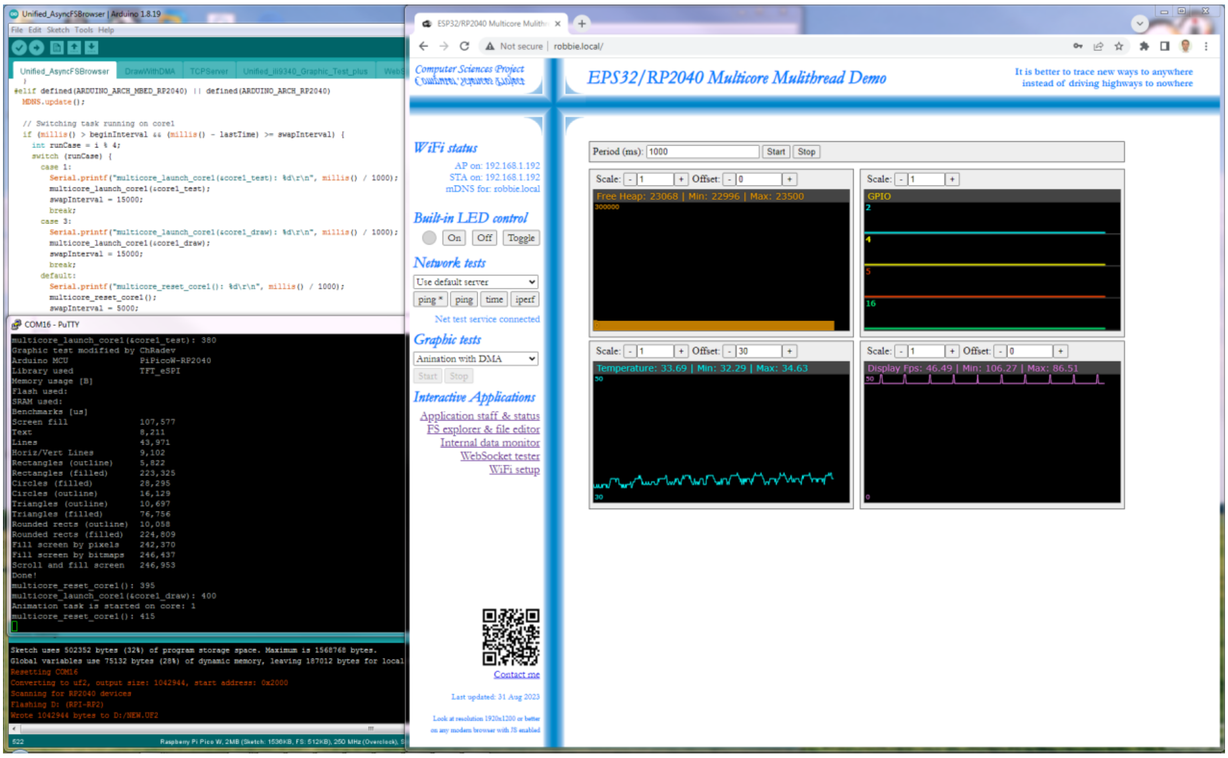
Save new copy of Unified multicore application under other name (like GPR_Display) and remove not relevant components like TCPServer.ino, DrawWithDMA.ino, Unified_ili9340_Graphic_Test_plus.ino and create new components GPR_Task.ino and GPS_Task.ino. Additional component SPIFFS based FSEditor.ino is added instead of SPIFFSEditor to overcome its problem if ESP32 core ver. 2.0 and later is used. Restructure SPIFFS file staff to separate private and public files. Restart Arduino IDE and re-flash SPIFFS changes to take effect. Modify the staff in GPR_Display.ino and WebSockets.ino according to changes made. Write required staff in GPR_Task.ino to read data from Serial2 and display them on 3.2” TFT SPI 240x320 display. GPR_Task has to be set to run on CPU core0 (not running main application). Use board configuration in Arduino -> Tools as shown above. ESP32-S3 may be programmed via native USB OTG JTAG/Serial port (COM23). Serial0 port via CH230 USB-to-Serial adapter (COM20) is used for debug information printing and as a spare channel for application programming. ESP32-S3 Serial2 port is used to connect GPR and Serial1 port is reserved to connect GPS module later on.

Web files are located in project subfolder named “data”. They can be flashed to internal flash SPIFFS using Arduino tool “ESP32 Sketch Data Upload”. Application “esp_tools_gui” may also be used to check ESP32-S3 information and configuration. For testing purposes JavaScript simulator is written to read GPR data from SEGY file and send them to ESP32-S3 Serial2 port (via CP2102 USB-to-Serial adapter on COM22). In future is planed JavaScript GPS simulator to be developed to read GPS data from file and send them to ESP32-S3 Serial1 port (via CP2102 USB-to-Serial adapter on COM21).

Notes:

- In future both USB OTG JTAG/Serial and CH230 USB to Serial0 ports may be used as USB Mass storage host (for storing of archive data files) and CDC device (for connecting to other computer) respectively but some issues have to be solved so currently they will be used for development purposes only;
- In case of more space needed SPIFFS partition scheme can be changed or ESP32-S3 module with more flash (16/32MB) can be used;
- Usage of SPI SD card slot located at 3.2” TFT SPI 240x320 display board for storing of archive files is also possible;
- Lack of ESP32-S3 reset after application flashing via USB OTG JTAG/Serial port bug is not observed;

Unified multicore application for ESP32 and RP2040 – more than combination of AsyncFSWebBrowser and graphic tests


Unified_AsyncFSBrowser | Arduino 1.8.19
File Edit Sketch Tools Help
Unified_AsyncFSBrowser
DrawWithDMA
TCPServer
Unified_ili9340_Graphic_Test_plus

```
  }
#elif defined(ARDUINO_ARCH_MBED_RP2040) || defined(ARDUINO_ARCH_RP2040)
  MDNS.update();

  // Switching task running on core1
  if (millis() > beginInterval && (millis() - lastTime) >= swapInterval) {
    int runCase = i % 4;
    switch (runCase) {
      case 1:
        Serial.printf("multicore_launch_core1(&core1_test): %d\r\n", millis() / 1000);
        multicore_launch_core1(&core1_test);
        swapInterval = 15000;
        break;
      case 3:
        Serial.printf("multicore_launch_core1(&core1_draw): %d\r\n", millis() / 1000);
        multicore_launch_core1(&core1_draw);
        swapInterval = 15000;
        break;
      default:
        Serial.printf("multicore_reset_core1(): %d\r\n", millis() / 1000);
        multicore_reset_core1();
        swapInterval = 5000;
```

COM16 - PuTTY

```
multicore_launch_core1(&core1_test): 380
Graphic test modified by ChRadev
Arduino MCU             PiPicoW-RP2040
Library used            TFT_eSPI
Memory usage [B]
Flash used:
SRAM used:
Benchmarks [us]
Screen fill             107,577
Text                    8,211
Lines                   43,971
Horiz/Vert Lines        9,102
Rectangles (outline)    5,822
Rectangles (filled)     223,325
Circles (filled)        28,295
Circles (outline)       16,129
Triangles (outline)     10,697
Triangles (filled)      76,756
Rounded rects (outline) 10,058
Rounded rects (filled)  224,809
Fill screen by pixels   242,370
Fill screen by bitmaps  246,437
Scroll and fill screen  246,953
Done!
multicore_reset_core1(): 395
multicore_launch_core1(&core1_draw): 400
Animation task is started on core: 1
multicore_reset_core1(): 415
```

```
Sketch uses 502352 bytes (32%) of program storage space. Maximum is 1568768 bytes.
Global variables use 75132 bytes (28%) of dynamic memory, leaving 187012 bytes for local
Resetting COM16
Converting to uf2, output size: 1042944, start address: 0x2000
Scanning for RP2040 devices
Flashing D: (RPI-RP2)
Wrote 1042944 bytes to D:/NEW.UF2
```

522   Raspberry Pi Pico W, 2MB (Sketch: 1536KB, FS: 512KB), 250 MHz (Overclock), S

ESP32/RP2040 Multicore Mulithr
Not secure | robbie.local/

Computer Sciences Project

EPS32/RP2040 Multicore Mulithread Demo

It is better to trace new ways to anywhere instead of driving highways to nowhere

WiFi status
AP on: 192.168.1.192
STA on: 192.168.1.192
mDNS for: robbie.local

Built-in LED control
○ On | Off | Toggle

Network tests
Use default server
ping * | ping | time | iperf
Net test service connected

Graphic tests
Anination with DMA
Start | Stop

Interactive Applications
Application staff & status
FS explorer & file editor
Internal data monitor
WebSocket tester
WiFi setup

Contact me
Last updated: 31 Aug 2023
Look at resolution 1920x1200 or better on any modern browser with JS enabled

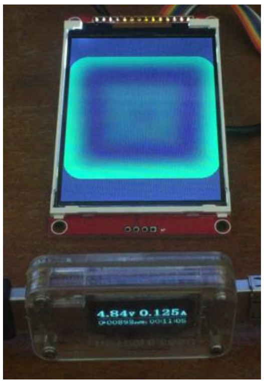
Period (ms): 1000 | Start | Stop

Scale: - 1 + Offset: - 0 +
Free Heap: 23068 | Min: 22996 | Max: 23500

Scale: - 1 +
GPIO

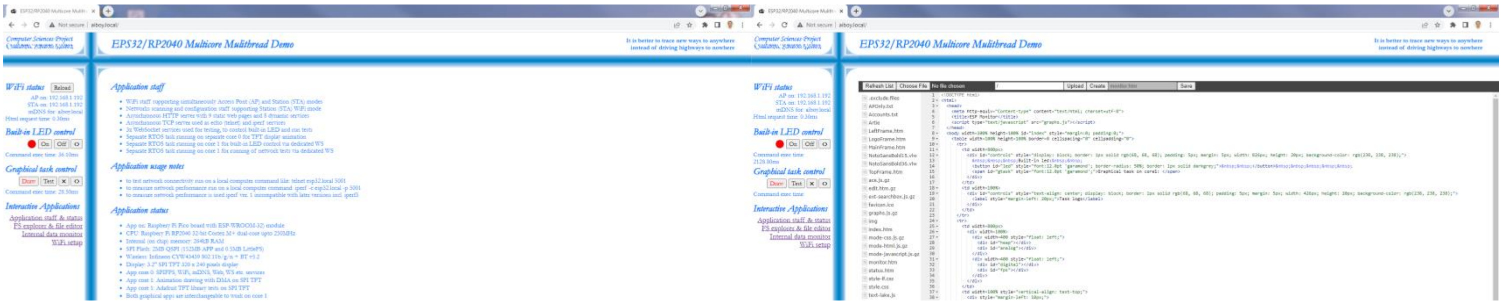
Scale: - 1 + Offset: - 30 +
Temperature: 33.69 | Min: 32.29 | Max: 34.63

Scale: - 1 + Offset: - 0 +
Display Fps: 46.49 | Min: 106.27 | Max: 86.51



Unified multicore application is based on AsyncFSBrowser, Unified graphic test (TFT-eSPI library) and DrawWithDMA sketches. It is Arduino IDE multi-file project for ESP32-WROOM and RP2040 based boards. It is running on both Arduino D1 R32 ESP32 (ESP32-WROOM) and Raspberry Pi Pco W (RP2040) boards and implements web server and sockets, console with printout of the Adafruit TFT tests adapted for eTFT library and log of swapping graphical tasks to work on the other core, web application with internal monitor showing graphs of the free heap memory, the GPIO states, the internal temperature sensor and the animation frame rate. On the right are shown pictures of graphical tasks (animation and graphical tests) running alternatively on the other core.

The most attractive application feature is almost complete independency of the performance of tasks running on different CPU cores. Other impressive result is graphical performance of animation task (46 frames per second at 42 bouncing circles) and all Adafruit TFT tests adapted to work with the eTFT library (especially scrolling 320x240 graphics at speed of 0.8ms per 240 pixels line). Next pictures represent final version of unified multicore application in action.


EPS32/RP2040 Multicore Mulithread Demo
It is better to trace new ways to anywhere instead of driving highways to nowhere
WiFi status
Built-in LED control
Graphical task control
Interactive Applications
Application staff & status
FS explorer & file editor
Internal data monitor
WiFi setup

Application staff

- WiFi staff supporting simultaneously Access Point (AP) and Station (STA) modes
- Networks scanning and configuration staff supporting Station (STA) WiFi mode
- Asynchronous HTTP server with 9 static web pages and 8 dynamic services
- Asynchronous TCP server used as echo (telnet) and iperf services
- 3x WebSocket services used for testing, to control built-in LED and run tests
- Separate RTOS task running on separate core 0 for TFT display animation
- Separate RTOS task running on core 1 for built-in LED control via dedicated WS
- Separate RTOS task running on core 1 for running of network tests via dedicated WS

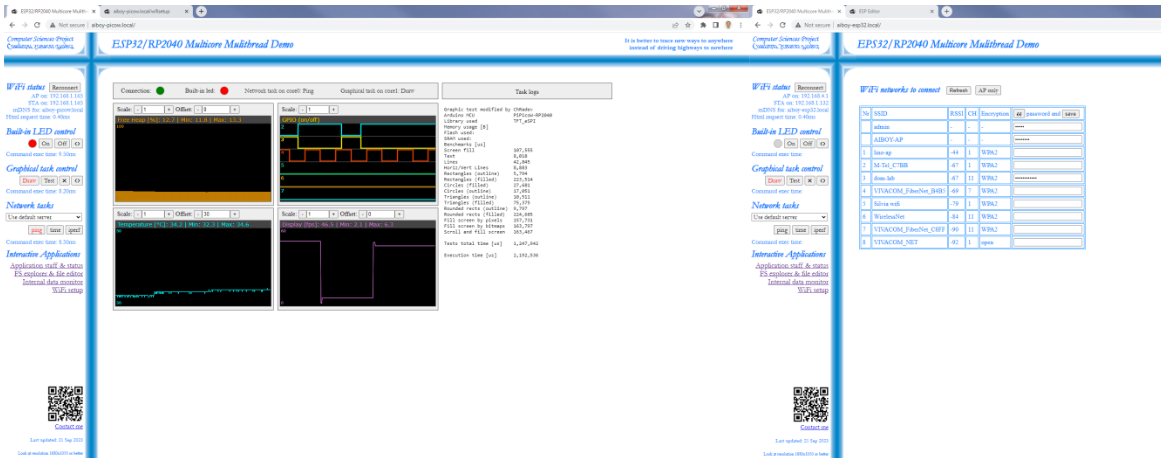
Application usage notes

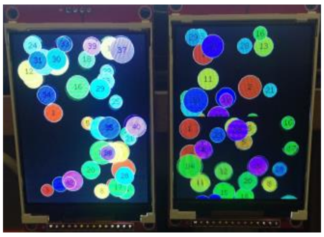
- to test network connectivity run on a local computer command like: telnet esp32.local 5001
- to measure network performance run on a local computer command: iperf -c esp32.local -p 5001
- to measure network performance is used iperf ver. 1 incompatible with later versions incl. iperf3

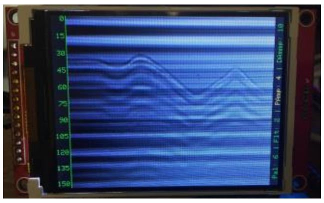
Application status

- App on: Raspberry Pi Pico board with ESP-WROOM-32) module
- CPU: Raspberry Pi RP2040 32-bit Cortex M+ dual-core upto 250MHz
- Internal (on chip) memory: 264kB RAM
- SPI Flash: 2MB QSPI (1.52MB APP and 0.5MB LittleFS)
- Wireless: Infineon CYW43439 802.11b/g/n + BT v5.2
- Display: 3.2" SPI TFT 320 x 240 pixels display
- App core 0: SPIFFS, WiFi, mDNS, Web, WS etc. services
- App core 1: Animation drawing with DMA on SPI TFT
- App core 1: Adafruit TFT library tests on SPI TFT
- Both graphical apps are interchangeable to work on core 1

Home view of the web application with control staff and system information (left) and control staff with internal FS file viewer and editor (right)







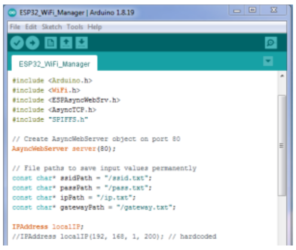
On the left most pictures are shown views with graphs of the internal system and application parameters (left) and WiFi setup page (right). It is evident that the results of the invoked commands and monitor (as well as other clients) are synchronized and ESP32 WiFi works in AP+STA mode (top left WiFi status). Graphical task log in the monitor view is showing the best results ever achieved (look at the table on page 7). This snapshot is taken after complete unification of the application for Pi Pico W (left) and ESP32 (right) platforms and adding network tasks control bar (only ping is implemented for the moment).

On the right most pictures are shown TFT displays in action connected to Pi Pico (top left) and ESP32 (top right) boards while running unified application. The current state of the application is not stable and has some bugs especially for ESP32 where bouncing circles are only moving at the bottom half of the screen, Adafruit tests do not work as expected and the application crashes frequently at exchanging of the graphic tasks. In case of Pi Pico applications is more stable but crashes form time to time at exchanging of the graphic tasks.

In the special modification of the unified multicore application running on ESP32-S3-R8N8 (right most bottom) is implemented preliminary test for displaying of data from GPR (Ground Penetration Radar) received via serial link and shown as 320x240 pixels scrolling graphics with up to 18 tracks per second speed.

Network performance using AsyncWebServer and AsyncTCP libraries on Pi Pico W and ESP32 series of boards

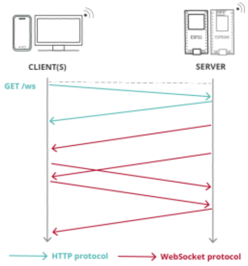
Startup projects working on ESP32 S2 Olimex boards and based on ESPAsyncWebServer library for Arduino:


ESP32_WiFi_Manager | Arduino 1.8.19
File Edit Sketch Tools Help
ESP32_WiFi_Manager

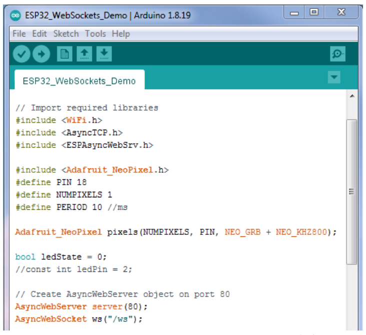
```
#include <Arduino.h>
#include <WiFi.h>
#include <ESPAsyncWebSrv.h>
#include <AsyncTCP.h>
#include "SPIFFS.h"

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

// File paths to save input values permanently
const char* ssidPath = "/ssid.txt";
const char* passPath = "/pass.txt";
const char* ipPath = "/ip.txt";
const char* gatewayPath = "/gateway.txt";

IPAddress localIP;
//IPAddress localIP(192, 168, 1, 200); // hardcoded
```

- ESP32: Create a Wi-Fi Manager (AsyncWebServer library)

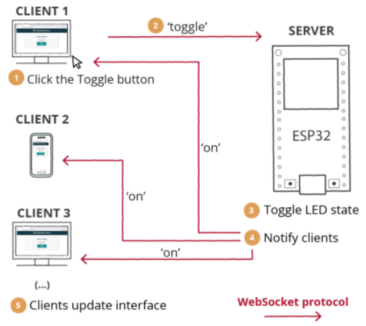
Application uses SPIFS on ESP32 systems to hold web and configuration files which have to be written manually by “ESP32 Sketch Data Upload” tool of Arduino IDE. The application first runs in AP mode asking for connection credentials of the local router. After storing them in FS files and restart it runs in STA mode. Main web page allows controlling built-in LED.



- ESP32 WebSocket Server: Control Outputs (Arduino IDE)


ESP32_WebSockets_Demo | Arduino 1.8.19
File Edit Sketch Tools Help
ESP32_WebSockets_Demo

```
// Import required libraries
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebSrv.h>

#include <Adafruit_NeoPixel.h>
#define PIN 18
#define NUMPIXELS 1
#define PERIOD 10 //ms

Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

bool ledState = 0;
//const int ledPin = 2;

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);
AsyncWebSocket ws("/ws");
```

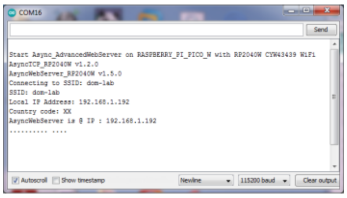


Application on ESP32 runs in STA mode with credentials defined in the sketch and open WebSocket server to control the LED. Its status can be changed by any client and will be updated at all the clients.

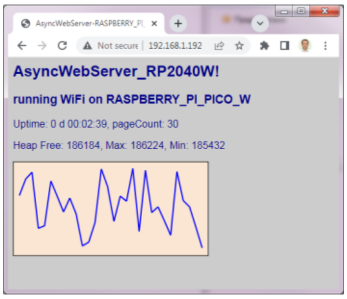
It was used Adafruit NeoPixel library to run above projects on Olimex ESP32 S2 series of boards with RGB instead of regular LED.

Startup projects on Raspberry Pi Pico W – AsyncWebServer for RP2040W library examples:

Async_AdvancedWebServer | Arduino 1.8.19
File Edit Sketch Tools Help
Async_AdvancedWebServer



```
Start Async_AdvancedWebServer on RASPBERRY_PI_PICO_W with RP2040W CYW43439 WiFi
AsyncTCP_RP2040W v1.2.0
AsyncWebServer_RP2040W v1.5.0
Connecting to SSID: dom-lab
SSID: dom-lab
Local IP Address: 192.168.1.192
Country code: XX
AsyncWebServer is @ IP : 192.168.1.192
.......... ....
```



AsyncWebServer_RP2040W!

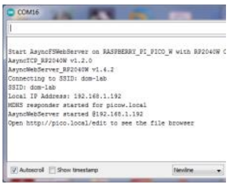
running WiFi on RASPBERRY_PI_PICO_W

Uptime: 0 d 00:02:39, pageCount: 30

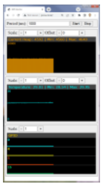
Heap Free: 186184, Max: 186224, Min: 185432

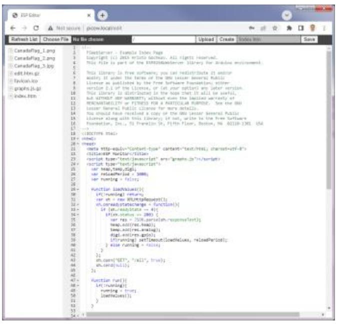
- Async_AdvancedWebServer

AsyncFSWebServer | Arduino 1.8.19
File Edit Sketch Tools Help
AsyncFSWebServer



```
Start AsyncFSWebServer on RASPBERRY_PI_PICO_W with RP2040W C
AsyncTCP_RP2040W v1.2.0
AsyncWebServer_RP2040W v1.4.2
Connecting to SSID: dom-lab
SSID: dom-lab
Local IP Address: 192.168.1.192
MDNS responder started for picow.local
AsyncWebServer started @192.168.1.192
Open http://pico.local/edit to see the file browser
```

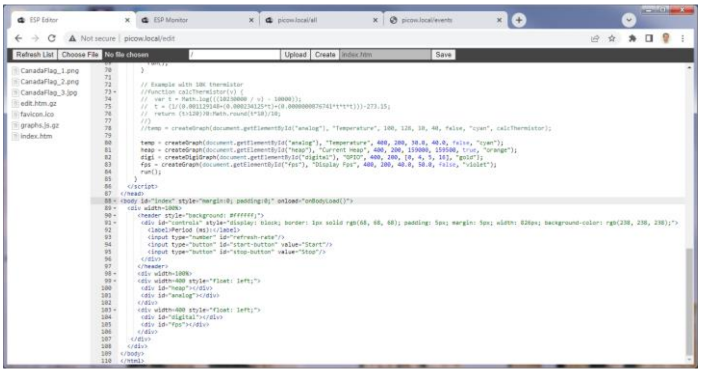




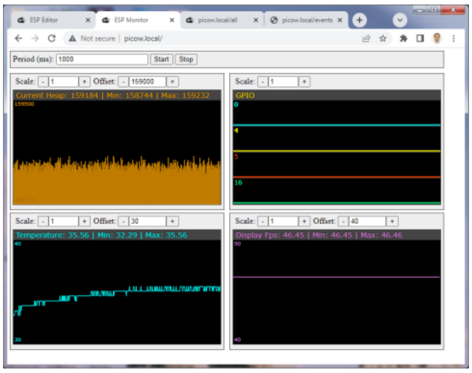
- AsyncFSWebServer (library ver. 1.5.0 did not run out of the box)

Application uses LittleFS library to access SPI flash FS. It also uses mDNS, basic authentication, AsyncWebSocket, AsyncEventSource and AsyncFSEditor_RP2040W library to show and edit files.

DrawWithDMA TFT_eSPI library test was compiled and run successfully on Raspberry Pi Pico W. Later on AsyncFSWebServer and DrawWithDMA combined multicore application was done by simply putting both files in a single project, renaming setup and loop functions in the second file to setup1 and loop1 and commenting the line Serial.begin(115200). Display drawing (42 circles) speed was the same (17.85fps) without appreciable change in the web access. Temperature measured by internal sensor is increased with approximately 2°C (up to 31°C). The heap is increased from 5kB up to 159kB. CPU overclocking to 250MHz did not speed up display drawing and web access but increase the temperature with approximately 3°C (up to 34°C). SPI speed can be changed in User_Setup.h of TFT_eSPI library. Changing it from 27MHz to 55MHz (2x) did not speed up display drawing but thanks to Bodmer comment and CPU clocking at 125MHz (SPI clock is 62.5MHz) display drawing can be speed up to 43-45fps @42 circles and 46.3fps @36 circles. Overclocking CPU to 250MHz (probably SPI clock is again 62.5MHz) increase display drawing speed up to 46.5fps @42 circles (2.6x) while working smoothly and reliably. Total consumption is increased form 110mA in case of overclocked DrawWithDMA single core application up to 144mA for combined multicore application.



+



+

Remote file manager and editor

On-line monitor

SPI TFT display

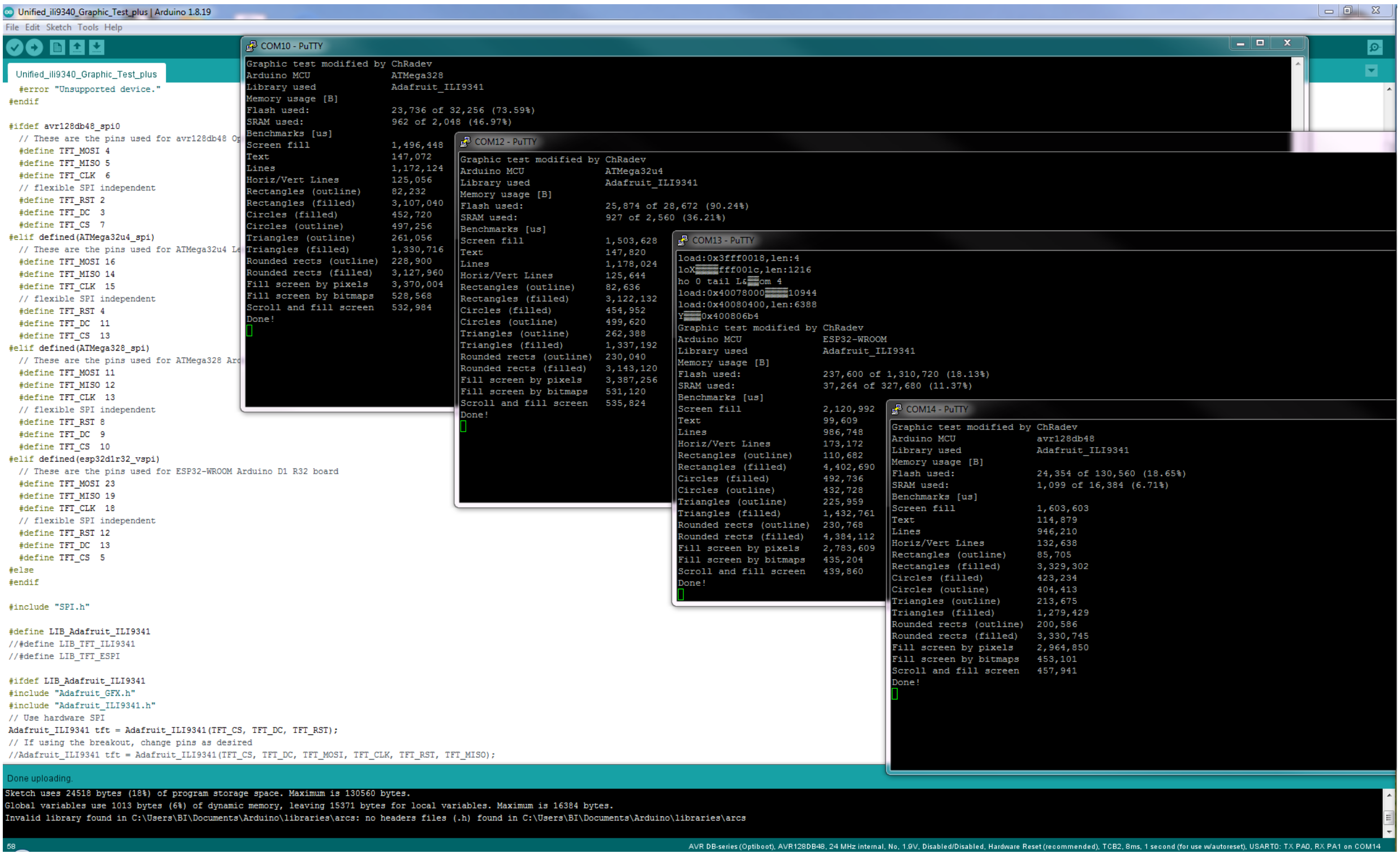
AsyncWebServer for RP2040W library built by Khoi Hoang is based on and modified from ESPAsyncWebServer library support of ESP32 and ESP8266 on Arduino cores. Next steps to be done for building of unified multicore application:

- Check of the code compatibility for both ESP32 and Pi Pico W boards;
- Dynamically running of different tasks on the second CPU core;
- Build unified web server application with WiFi working in AP and/or STA modes including its management, mDNS, LittleFS, WebSockets etc.

Connection setup for 3.2” 240x320 pixels TFT display with SPI interface

	3.2” TFT SPI LCD Display	Arduino UNO ATMega328	Olimexino32U4 ATMega32u4	Optiboot AVR128db48	Arduino R32 ESP-WROOM-32	Raspberry PI Pico RP2040	ESP32-S3-WROOM	Signal description (3.2” TFT SPI LCD Display)
1	VCC	VCC-3.3V	VCC-3.3V	VCC-3.3V	VCC-3.3V	VCC-3.3V	3.3V	3.3V power input (do not connect to 5V)
2	GND	GND	GND	GND	GND	GND	GND	GND
3	CS	D10	D13	0,#SS, PA7	IO05	GP17	GPIO10	LCD chip select signal, low level enable
4	RESET	D8	D4	PA2 (0,SDA)	IO12	GP21	GPIO9	LCD reset signal, low level reset
5	DC/RS	D9	D11	PA3 (0,SCL)	IO13	GP20	GPIO14	LCD register / data selection signal, high level: register, low level: data
6	SDI(MOSI)	D11	D16	0,MOSI, PA4	IO23	GP16	GPIO11	SPI bus write data signal
7	SCK	D13	D15	0,SCK, PA6	IO18	GP18	GPIO12	SPI bus clock signal
8	LED	VCC-5V	VCC-5V	VCC-5V	VCC-5V	VCC-5V	5V	Backlight control, high level lighting, if not controlled, connect 5V for always bright
9	SDO(MISO)	D12	D14	0,MISO, PA5	IO19	GP19	GPIO13	SPI bus read data signal, if you do not need to the read function, you cannot connect it

All 4 boards are connected to 3.2”SPI TFT display and running Unified graphic test


Unified_ili9340_Graphic_Test_plus | Arduino 1.8.19
File Edit Sketch Tools Help
Unified_ili9340_Graphic_Test_plus
COM10 - PuTTY
COM12 - PuTTY
COM13 - PuTTY
COM14 - PuTTY

```
  #error "Unsupported device."
#endif

#ifdef avr128db48_spi0
  // These are the pins used for avr128db48 Op
  #define TFT_MOSI 4
  #define TFT_MISO 5
  #define TFT_CLK  6
  // flexible SPI independent
  #define TFT_RST 2
  #define TFT_DC  3
  #define TFT_CS  7
#elif defined(ATMega32u4_spi)
  // These are the pins used for ATMega32u4 Le
  #define TFT_MOSI 16
  #define TFT_MISO 14
  #define TFT_CLK  15
  // flexible SPI independent
  #define TFT_RST 4
  #define TFT_DC  11
  #define TFT_CS  13
#elif defined(ATMega328_spi)
  // These are the pins used for ATMega328 Ard
  #define TFT_MOSI 11
  #define TFT_MISO 12
  #define TFT_CLK  13
  // flexible SPI independent
  #define TFT_RST 8
  #define TFT_DC  9
  #define TFT_CS  10
#elif defined(esp32d1r32_vspi)
  // These are the pins used for ESP32-WROOM Arduino D1 R32 board
  #define TFT_MOSI 23
  #define TFT_MISO 19
  #define TFT_CLK  18
  // flexible SPI independent
  #define TFT_RST 12
  #define TFT_DC  13
  #define TFT_CS  5
#else
#endif

#include "SPI.h"

#define LIB_Adafruit_ILI9341
//#define LIB_TFT_ILI9341
//#define LIB_TFT_ESPI

#ifdef LIB_Adafruit_ILI9341
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9341.h"
// Use hardware SPI
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_RST);
// If using the breakout, change pins as desired
//Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_MOSI, TFT_CLK, TFT_RST, TFT_MISO);
```

COM10 - PuTTY

```
Graphic test modified by ChRadev
Arduino MCU              ATMega328
Library used             Adafruit_ILI9341
Memory usage [B]
Flash used:              23,736 of 32,256 (73.59%)
SRAM used:               962 of 2,048 (46.97%)
Benchmarks [us]
Screen fill              1,496,448
Text                     147,072
Lines                    1,172,124
Horiz/Vert Lines         125,056
Rectangles (outline)     82,232
Rectangles (filled)      3,107,040
Circles (filled)         452,720
Circles (outline)        497,256
Triangles (outline)      261,056
Triangles (filled)       1,330,716
Rounded rects (outline)  228,900
Rounded rects (filled)   3,127,960
Fill screen by pixels    3,370,004
Fill screen by bitmaps   528,568
Scroll and fill screen   532,984
Done!
```

COM12 - PuTTY

```
Graphic test modified by ChRadev
Arduino MCU              ATMega32u4
Library used             Adafruit_ILI9341
Memory usage [B]
Flash used:              25,874 of 28,672 (90.24%)
SRAM used:               927 of 2,560 (36.21%)
Benchmarks [us]
Screen fill              1,503,628
Text                     147,820
Lines                    1,178,024
Horiz/Vert Lines         125,644
Rectangles (outline)     82,636
Rectangles (filled)      3,122,132
Circles (filled)         454,952
Circles (outline)        499,620
Triangles (outline)      262,388
Triangles (filled)       1,337,192
Rounded rects (outline)  230,040
Rounded rects (filled)   3,143,120
Fill screen by pixels    3,387,256
Fill screen by bitmaps   531,120
Scroll and fill screen   535,824
Done!
```

COM13 - PuTTY

```
load:0x3fff0018,len:4
loX▒▒▒fff001c,len:1216
ho 0 tail L&▒om 4
load:0x40078000▒▒▒10944
load:0x40080400,len:6388
Y▒▒0x400806b4
Graphic test modified by ChRadev
Arduino MCU              ESP32-WROOM
Library used             Adafruit_ILI9341
Memory usage [B]
Flash used:              237,600 of 1,310,720 (18.13%)
SRAM used:               37,264 of 327,680 (11.37%)
Benchmarks [us]
Screen fill              2,120,992
Text                     99,609
Lines                    986,748
Horiz/Vert Lines         173,172
Rectangles (outline)     110,682
Rectangles (filled)      4,402,690
Circles (filled)         492,736
Circles (outline)        432,728
Triangles (outline)      225,959
Triangles (filled)       1,432,761
Rounded rects (outline)  230,768
Rounded rects (filled)   4,384,112
Fill screen by pixels    2,783,609
Fill screen by bitmaps   435,204
Scroll and fill screen   439,860
Done!
```

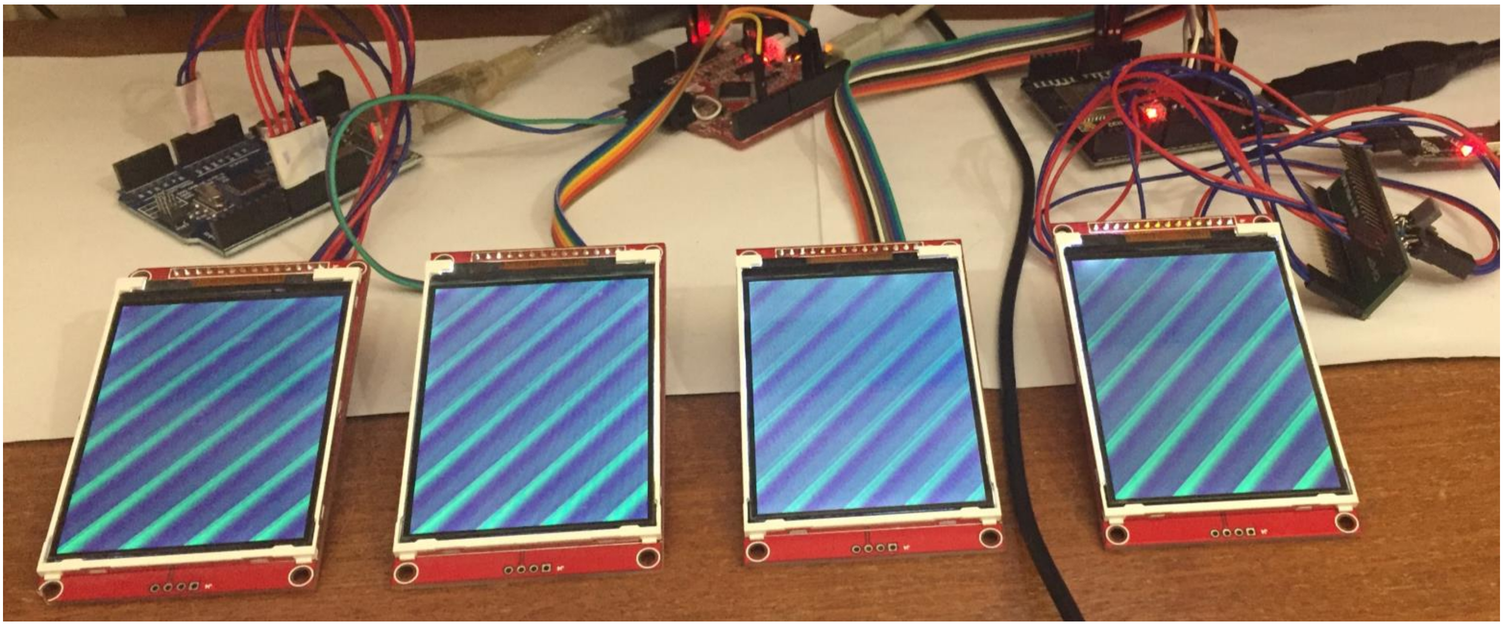
COM14 - PuTTY

```
Graphic test modified by ChRadev
Arduino MCU              avr128db48
Library used             Adafruit_ILI9341
Memory usage [B]
Flash used:              24,354 of 130,560 (18.65%)
SRAM used:               1,099 of 16,384 (6.71%)
Benchmarks [us]
Screen fill              1,603,603
Text                     114,879
Lines                    946,210
Horiz/Vert Lines         132,638
Rectangles (outline)     85,705
Rectangles (filled)      3,329,302
Circles (filled)         423,234
Circles (outline)        404,413
Triangles (outline)      213,675
Triangles (filled)       1,279,429
Rounded rects (outline)  200,586
Rounded rects (filled)   3,330,745
Fill screen by pixels    2,964,850
Fill screen by bitmaps   453,101
Scroll and fill screen   457,941
Done!
```

Done uploading.

```
Sketch uses 24518 bytes (18%) of program storage space. Maximum is 130560 bytes.
Global variables use 1013 bytes (6%) of dynamic memory, leaving 15371 bytes for local variables. Maximum is 16384 bytes.
Invalid library found in C:\Users\BI\Documents\Arduino\libraries\arcs: no headers files (.h) found in C:\Users\BI\Documents\Arduino\libraries\arcs
```

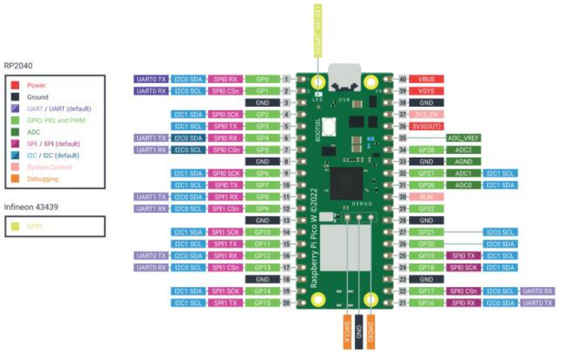
58 AVR DB-series (Optiboot), AVR128DB48, 24 MHz internal, No, 1.9V, Disabled/Disabled, Hardware Reset (recommended), TCB2, 8ms, 1 second (for use w/autoreset), USART0: TX PA0, RX PA1 on COM14



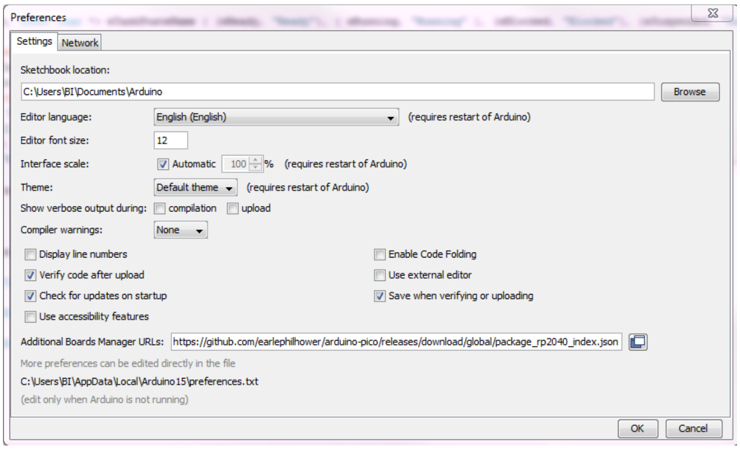
Arduino UNO (ATMega328)   Olimexino-32U4 (ATMega32u4)   Arduino D1 R32 (ESP32)   Optiboot (AVR128db48)

Arduino Pi Pico (W) boards

- For using Pi Pico (W) boards



- In Preferences add URL: https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json



Preferences

Settings | Network

Sketchbook location: C:\Users\BI\Documents\Arduino [Browse]

Editor language: English (English) (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: □ compilation □ upload

Compiler warnings: None

□ Display line numbers

☑ Verify code after upload

☑ Check for updates on startup

□ Use accessibility features

□ Enable Code Folding

□ Use external editor

☑ Save when verifying or uploading

Additional Boards Manager URLs: https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json

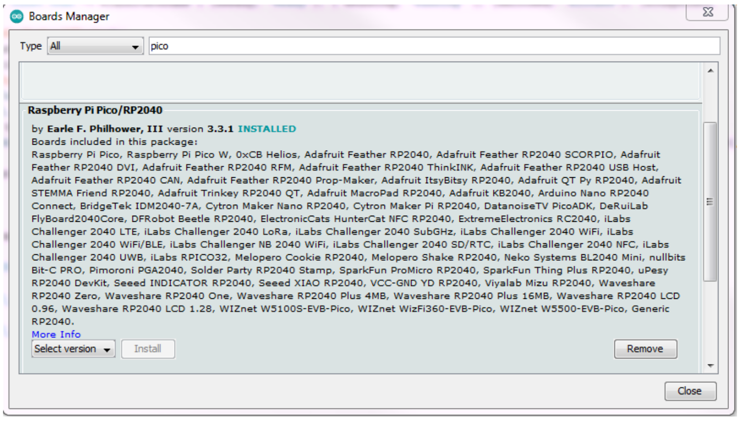
More preferences can be edited directly in the file

C:\Users\BI\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

[OK] [Cancel]

- Install Pi Pico / RP2040 in board manager



Boards Manager

Type: All pico

Raspberry Pi Pico/RP2040

by **Earle F. Philhower, III** version **3.3.1** INSTALLED

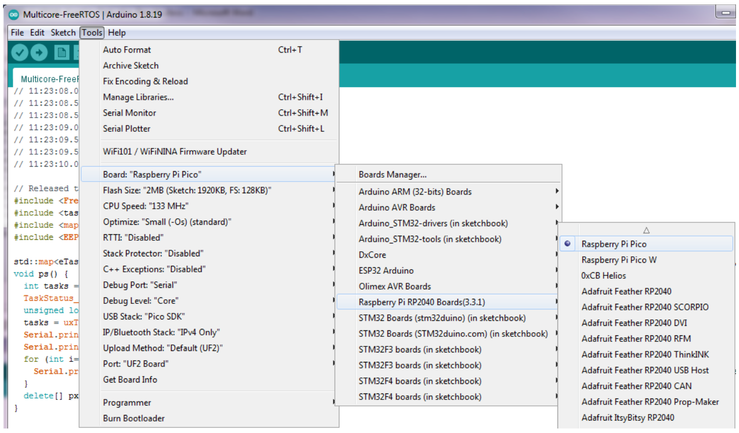
Boards included in this package:

Raspberry Pi Pico, Raspberry Pi Pico W, 0xCB Helios, Adafruit Feather RP2040, Adafruit Feather RP2040 SCORPIO, Adafruit Feather RP2040 DVI, Adafruit Feather RP2040 RFM, Adafruit Feather RP2040 ThinkINK, Adafruit Feather RP2040 USB Host, Adafruit Feather RP2040 CAN, Adafruit Feather RP2040 Prop-Maker, Adafruit ItsyBitsy RP2040, Adafruit QT Py RP2040, Adafruit STEMMA Friend RP2040, Adafruit Trinkey RP2040 QT, Adafruit MacroPad RP2040, Adafruit KB2040, Arduino Nano RP2040 Connect, BridgeTek IDM2040-7A, Cytron Maker Nano RP2040, Cytron Maker Pi RP2040, DatanoiseTV PicoADK, DeRuiLab FlyBoard2040Core, DFRobot Beetle RP2040, ElectronicCats HunterCat NFC RP2040, ExtremeElectronics RC2040, iLabs Challenger 2040 LTE, iLabs Challenger 2040 LoRa, iLabs Challenger 2040 SubGHz, iLabs Challenger 2040 WiFi, iLabs Challenger 2040 WiFi/BLE, iLabs Challenger NB 2040 WiFi, iLabs Challenger 2040 SD/RTC, iLabs Challenger 2040 NFC, iLabs Challenger 2040 UWB, iLabs RPICO32, Melopero Cookie RP2040, Melopero Shake RP2040, Neko Systems BL2040 Mini, nullbits Bit-C PRO, Pimoroni PGA2040, Solder Party RP2040 Stamp, SparkFun ProMicro RP2040, SparkFun Thing Plus RP2040, uPesy RP2040 DevKit, Seeed INDICATOR RP2040, Seeed XIAO RP2040, VCC-GND YD RP2040, Viyalab Mizu RP2040, Waveshare RP2040 Zero, Waveshare RP2040 One, Waveshare RP2040 Plus 4MB, Waveshare RP2040 Plus 16MB, Waveshare RP2040 LCD 0.96, Waveshare RP2040 LCD 1.28, WIZnet W5100S-EVB-Pico, WIZnet WizFi360-EVB-Pico, WIZnet W5500-EVB-Pico, Generic RP2040.

More Info

[Select version] [Install] [Remove] [Close]

- Install "Raspberry Pi Pico" or "Raspberry Pi Pico W" board



Multicore-FreeRTOS | Arduino 1.8.19

File Edit Sketch Tools Help

Board: "Raspberry Pi Pico"

Flash Size: "2MB (Sketch: 1920KB, FS: 128KB)"

CPU Speed: "133 MHz"

Optimize: "Small (-Os) (standard)"

RTTI: "Disabled"

Stack Protector: "Disabled"

C++ Exceptions: "Disabled"

Debug Port: "Serial"

Debug Level: "Core"

USB Stack: "Pico SDK"

IP/Bluetooth Stack: "IPv4 Only"

Upload Method: "Default (UF2)"

Port: "UF2 Board"

Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Programmer
Burn Bootloader
Boards Manager...
Arduino ARM (32-bits) Boards
Arduino AVR Boards
Arduino_STM32-drivers (in sketchbook)
Arduino_STM32-tools (in sketchbook)
DxCore
ESP32 Arduino
Olimex AVR Boards
Raspberry Pi RP2040 Boards(3.3.1)
STM32 Boards (stm32duino) (in sketchbook)
STM32 Boards (STM32duino.com) (in sketchbook)
STM32F3 boards (in sketchbook)
STM32F3 boards (in sketchbook)
STM32F4 boards (in sketchbook)
STM32F4 boards (in sketchbook)
● Raspberry Pi Pico
Raspberry Pi Pico W
0xCB Helios
Adafruit Feather RP2040
Adafruit Feather RP2040 SCORPIO
Adafruit Feather RP2040 DVI
Adafruit Feather RP2040 RFM
Adafruit Feather RP2040 ThinkINK
Adafruit Feather RP2040 USB Host
Adafruit Feather RP2040 CAN
Adafruit Feather RP2040 Prop-Maker
Adafruit ItsyBitsy RP2040

```
Multicore-FreeRTOS
// 11:23:08.0
// 11:23:08.5
// 11:23:08.5
// 11:23:09.0
// 11:23:09.5
// 11:23:09.5
// 11:23:10.0

// Released t
#include <Fre
#include <tas
#include <map
#include <EEP

std::map<eTas
void ps() {
  int tasks =
  TaskStatus_
  unsigned lo
  tasks = uxT
  Serial.prin
  Serial.prin
  for (int i=
    Serial.pr
  }
  delete[] px
}
```

- Connect the board to Windows PC while BOOTSEL button is pushed – "RPI-RP2" mass storage device should be appeared
- After uploading the sketch "Pico" or "Pico W" device will be appeared in "Device Manager"
- Update its device driver using Atmel USB to serial INF file changing [DeviceList.*] sections to:
  %PI_CDC_PICO%=DriverInstall, USB\VID_2E8A&PID_000A&REV_0100 or
  %PI_CDC_PICO%=DriverInstall, USB\VID_2E8A&PID_F00A&REV_0100
- Change [Strings] sections also to appropriate once

Multicore version of "Hello World and Blinking LED" common test for Pi Pico

- Open from File -> Examples -> (Examples for Paspberry Pi Pico) -> FreeRTOS -> Milticore FreeRTOS sketch and safe it in your Arduino sketch folder:

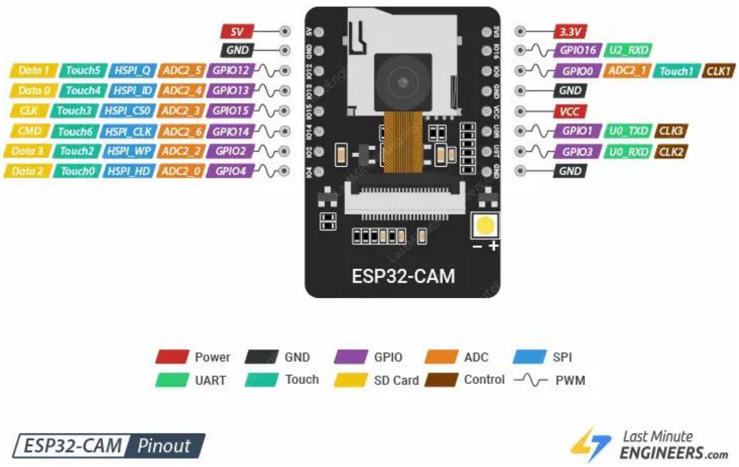
```
#include <FreeRTOS.h>
#include <task.h>
#include <map>
#include <EEPROM.h>
std::map<eTaskState, const char *> eTaskStateName {
{eReady, "Ready"}, { eRunning, "Running" }, {eBlocked,
"Blocked"}, {eSuspended, "Suspended"}, {eDeleted,
"Deleted"} };
void ps() {
  int tasks = uxTaskGetNumberOfTasks();
  TaskStatus_t *pxTaskStatusArray = new
TaskStatus_t[tasks];
  unsigned long runtime;
  tasks = uxTaskGetSystemState( pxTaskStatusArray, tasks,
&runtime );
  Serial.printf("# Tasks: %d\r\n", tasks);
  Serial.println("ID, NAME, STATE, PRIO, CYCLES");
  for (int i=0; i < tasks; i++) {
    Serial.printf("%d: %-16s %-10s %d %lu\r\n", i,
pxTaskStatusArray[i].pcTaskName,
eTaskStateName[pxTaskStatusArray[i].eCurrentState],
(int)pxTaskStatusArray[i].uxCurrentPriority,
pxTaskStatusArray[i].ulRunTimeCounter);
  }
  delete[] pxTaskStatusArray;
}
void blink(void *param) {
   (void) param;
  pinMode(LED_BUILTIN, OUTPUT);
  while (true) {
    digitalWrite(LED_BUILTIN, LOW);
    delay(750);
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
  }
}
void setup() {
  Serial.begin(115200);
  xTaskCreate(blink, "BLINK", 128, nullptr, 1, nullptr);
  delay(5000);
}
volatile int val= 0;
void loop() {
  Serial.printf("C0: Blue leader standing by...\r\n");
  ps();
  Serial.printf("val: %d\r\n", val);
  delay(1000);
}
// Running on core1
void setup1() {
  delay(5000);
  Serial.printf("C1: Red leader standing by...\r\n");
}
void loop1() {
  static int x = 0;
  Serial.printf("C1: Stay on target...\r\n");
  val++;
  if (++x < 10) {
    EEPROM.begin(512);
    EEPROM.write(0,x);
    EEPROM.commit();
  }
  delay(1000);
}
```

- It demonstrates a simple use of the setup1()/loop1() functions for a multiprocessor run and following will be printed on the serial port while LED is blinking:

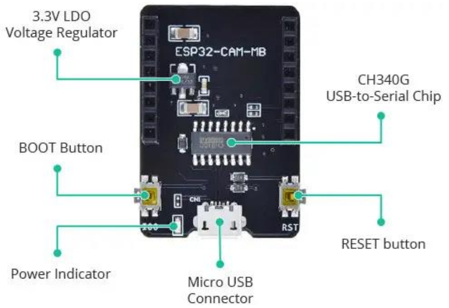
```
C1: Stay on target...
C0: Blue leader standing by...
# Tasks: 9
ID, NAME, STATE, PRIO, CYCLES
0: CORE0          Running    4 191473164
1: IDLE1          Running    0 3622023404
2: IDLE0          Ready      0 3371562651
3: BLINK          Blocked    1 5381238
4: CORE1          Blocked    4 103437988
5: USB            Blocked    6 3826967365
6: Tmr Svc        Blocked    2 21071
7: IdleCore1      Suspended  7 17213
8: IdleCore0      Suspended  7 88986822
val: 683
```

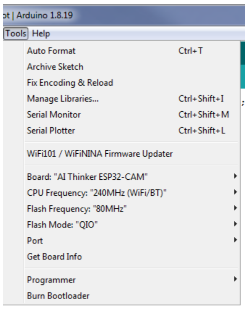
ESP32-CAM

- Getting Started With ESP32-CAM
- All examples work with ESP32 Espressif System 2.0.9


5V
GND
Data 1 Touch5 HSPI_Q ADC2_5 GPIO12
Data 0 Touch4 HSPI_ID ADC2_4 GPIO13
CLK Touch3 HSPI_CS0 ADC2_3 GPIO15
CMD Touch6 HSPI_CLK ADC2_6 GPIO14
Data 3 Touch2 HSPI_WP ADC2_2 GPIO2
Data 2 Touch0 HSPI_HD ADC2_0 GPIO4
3.3V
GPIO16 U2_RXD
GPIO0 ADC2_1 Touch1 CLK1
GND
VCC
GPIO1 U0_TXD CLK3
GPIO3 U0_RXD CLK2
GND
ESP32-CAM
Power GND GPIO ADC SPI
UART Touch SD Card Control PWM
ESP32-CAM Pinout
Last Minute ENGINEERS.com

- Using ESP32-CAM-MB module makes programming easy





Board: "AI Thinker ESP32-CAM"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"

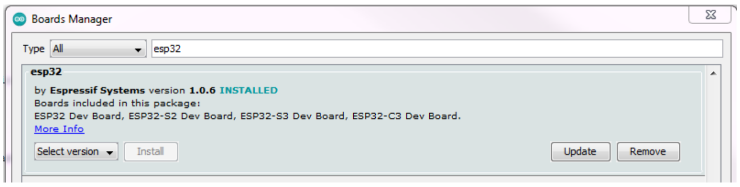
Wifi Camera Robot Car

- DIY ESP32 Camera Motor Shield - Wifi Camera Robot Car

https://www.olimex.com/Products/IoT/ESP32/ESP32-CAM/
https://www.instructables.com/DIY-ESP32-Camera-Motor-Shield-Wifi-Camera-Robot-Ca/,
https://dronebotworkshop.com/esp32-cam-intro/
https://randomnerdtutorials.com/esp32-cam-video-streaming-web-server-camera-home-assistant/
https://dronebotworkshop.com/esp32cam-robot-car/

- In “Resources” of the last link download: Code for ESP32CAM Car, the code needed to make this car work, all in one ZIP file.

- To compile it use ESP32 Espressif System 1.0.6



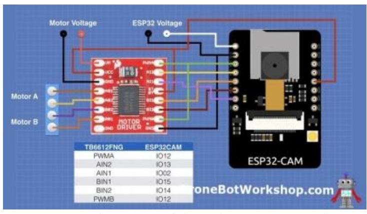
Boards Manager

Type All esp32

esp32
by **Espressif Systems** version **1.0.6** INSTALLED
Boards included in this package:
ESP32 Dev Board, ESP32-S2 Dev Board, ESP32-S3 Dev Board, ESP32-C3 Dev Board.
More Info

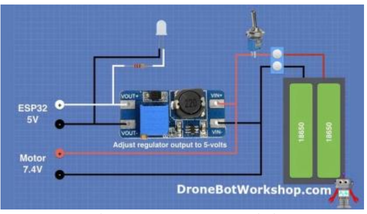
- Follow instructions in about the hardware : https://dronebotworkshop.com/esp32cam-robot-car/

- Main electrical parts



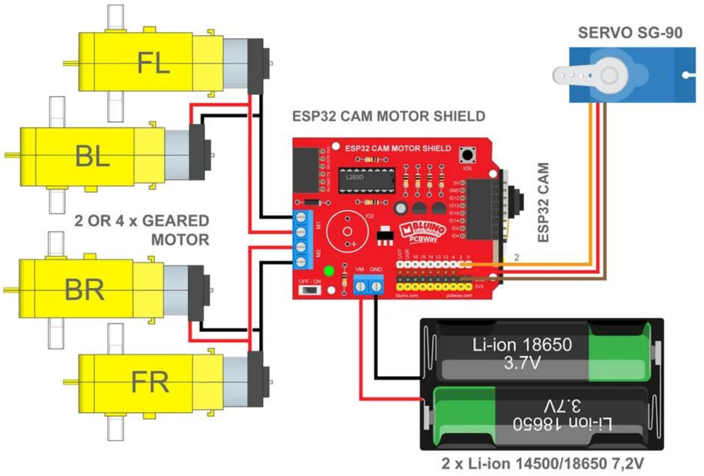
TB6612FNG	ESP32CAM
PWMA	IO12
AIN2	IO13
AIN1	IO02
BIN1	IO15
BIN2	IO14
PWMB	IO12

Camera and motor driver interconnection



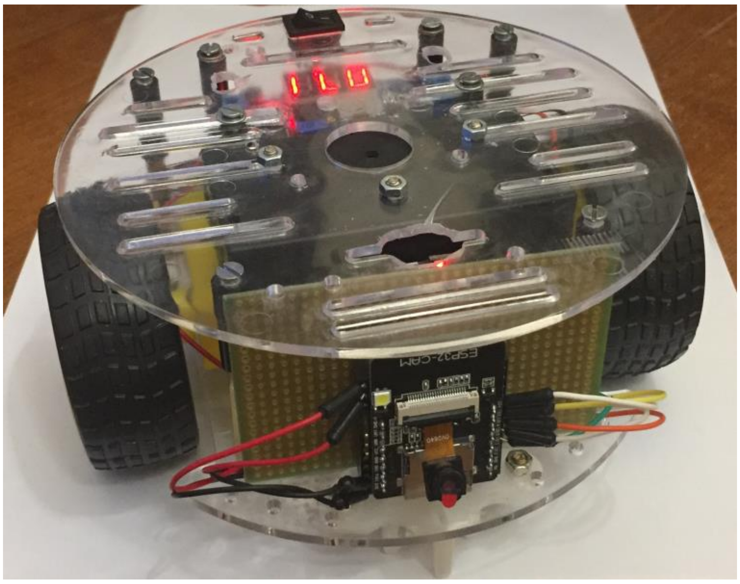
Motor and ESP32-CAM module power

- Power schematics



- Final results

Wifi Camera Robot Car – own implementation



ESP32-S2/3 boards on Arduino IDE

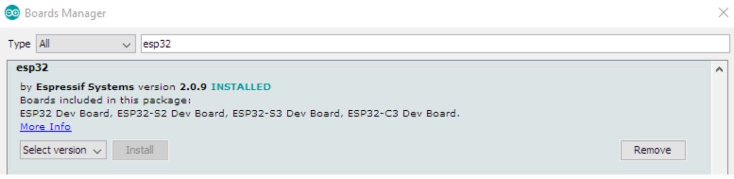
- For using ESP32-S2 boards like:

ESP32-S2-DevKit-Lipo

ESP32-S2-WROVER-DevKit-Lipo-USB

ESP32-S3-DevKit-Lipo

- Install the ESP32-S2 support for Arduino IDE
- In "File" → "Preferences" add URL: https://espressif.github.io/arduino-esp32/package_esp32_index.json
- In "Tools" → "Boards" → "Board Manager" search for the esp32 platform and install ver. 2.0.0 or later

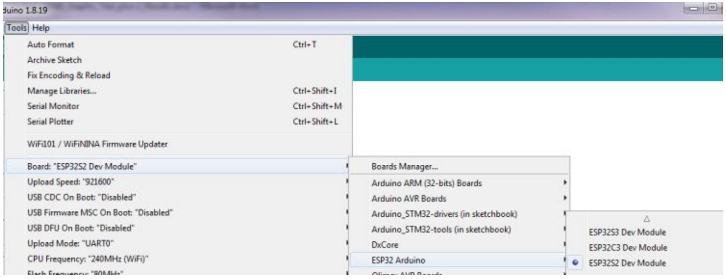

Boards Manager

Type All | esp32

esp32

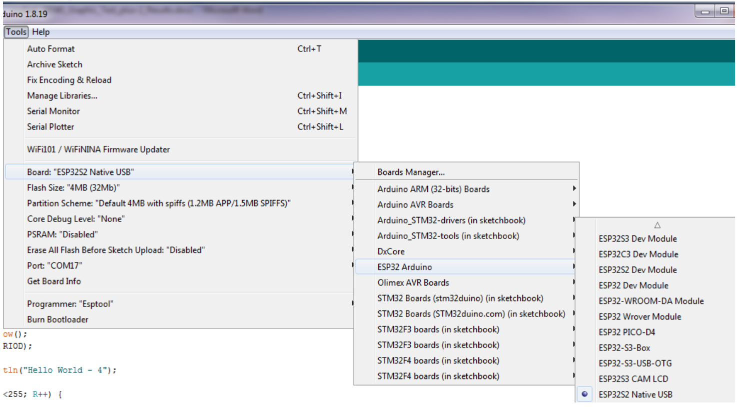
by Espressif Systems version 2.0.9 INSTALLED
Boards included in this package:
ESP32 Dev Board, ESP32-S2 Dev Board, ESP32-S3 Dev Board, ESP32-C3 Dev Board.
More Info

- Restart IDE and select board in "Tools" → "Board:
  - o "ESP32S2 Dev Module" for ESP32-S2-DevKit-Lipo


Board: "ESP32S2 Dev Module"
Upload Speed: "921600"
USB CDC On Boot: "Disabled"
USB Firmware MSC On Boot: "Disabled"
USB DFU On Boot: "Disabled"
Upload Mode: "UART0"
CPU Frequency: "240MHz (WiFi)"

duino 1.8.19
Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Boards Manager...
Arduino ARM (32-bits) Boards
Arduino AVR Boards
Arduino_STM32-drivers (in sketchbook)
Arduino_STM32-tools (in sketchbook)
DxCore
ESP32 Arduino
ESP32S3 Dev Module
ESP32C3 Dev Module
ESP32S2 Dev Module

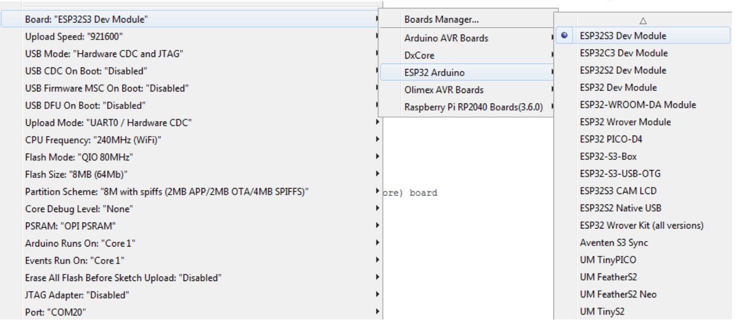
  - o "ESP32S2 Native USB" for ESP32-S2-WROVER-DevKit-Lipo-USB


Board: "ESP32S2 Native USB"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Erase All Flash Before Sketch Upload: "Disabled"
Port: "COM17"
Programmer: "Esptool"

duino 1.8.19
Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Burn Bootloader
Boards Manager...
Arduino ARM (32-bits) Boards
Arduino AVR Boards
Arduino_STM32-drivers (in sketchbook)
Arduino_STM32-tools (in sketchbook)
DxCore
ESP32 Arduino
Olimex AVR Boards
STM32 Boards (stm32duino) (in sketchbook)
STM32 Boards (STM32duino.com) (in sketchbook)
STM32F3 boards (in sketchbook)
STM32F3 boards (in sketchbook)
STM32F4 boards (in sketchbook)
STM32F4 boards (in sketchbook)
ESP32S3 Dev Module
ESP32C3 Dev Module
ESP32S2 Dev Module
ESP32 Dev Module
ESP32-WROOM-DA Module
ESP32 Wrover Module
ESP32 PICO-D4
ESP32-S3-Box
ESP32-S3-USB-OTG
ESP32S3 CAM LCD
ESP32S2 Native USB

```
ow();
RIOD);

tln("Hello World - 4");

<255; R++) {
```

  - o "ESP32S3 Dev Module" for ESP32-S3-DevKit-Lipo


Board: "ESP32S3 Dev Module"
Upload Speed: "921600"
USB Mode: "Hardware CDC and JTAG"
USB CDC On Boot: "Disabled"
USB Firmware MSC On Boot: "Disabled"
USB DFU On Boot: "Disabled"
Upload Mode: "UART0 / Hardware CDC"
CPU Frequency: "240MHz (WiFi)"
Flash Mode: "QIO 80MHz"
Flash Size: "8MB (64Mb)"
Partition Scheme: "8M with spiffs (2MB APP/2MB OTA/4MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "OPI PSRAM"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"
Erase All Flash Before Sketch Upload: "Disabled"
JTAG Adapter: "Disabled"
Port: "COM20"

- Connect ESP32-S2/3-DevKit-Lipo and install driver for USB-Serial CH340 adapter if needed
- Connect ESP32-S2-WROVER-DevKit-Lipo-USB and put it in boot loader's mode (hold GPIO0 low while reset)
- Install driver with Zadig software if needed
  - o Enable in "Options" → "List all devices"
  - o Choose device "ESP32-S2 (Interface 2)"
  - o And option "USB Serial (CDC)"
- Any time for programming ESP32-S2-WROVER-DevKit-Lipo-USB has to be put in boot loader's mode and reset manually after uploading the sketch

Multitasking "Hello World & RGB LED" test

```
/*
 * Requires Adafruit NeoPixel library
 */
#include <Adafruit_NeoPixel.h>
#define PIN 18
#define NUMPIXELS 1
#define PERIOD 10 //ms
Adafruit_NeoPixel pixels(NUMPIXELS, PIN,
                         NEO_GRB + NEO_KHZ800);
int colors[3];
void setup() {
  pixels.begin();
  for (int i = 0; i < 3; i++) colors[i] = 0;
#if 1
  Serial.begin(115200); // ESP32-S2-DevKit-Lipo
#else
  Serial.begin();// ESP32-S2-WROVER-DevKit-Lipo-USB
  // Wait for serial port to connect.
  // Needed for native USB port only.
  while (!Serial) ;
#endif
  Serial.println("Hello World!");
  vTaskDelay(1000 / portTICK_PERIOD_MS);
  xTaskCreate(loop2,"loop2", 2048, NULL,1,NULL);
}
int n = 0;
void loop2( void * parameter ) {
  while(1) {
    Serial.print("Hello World - "); Serial.println(n++);
    vTaskDelay(2000 / portTICK_PERIOD_MS);
  }
}
void loop () {
  for (int i = 0; i < 3; i++) {
    int j;
    for (j = 0; j < 256; j++) {
      colors[i] = j;
      pixels.setPixelColor(0, pixels.Color(colors[0],
                           colors[1], colors[2]));
      pixels.show(); delay (PERIOD);
    }
    for (j = 255; j >= 0; j--) {
      colors[i] = j;
      pixels.setPixelColor(0, pixels.Color(colors[0],
                           colors[1], colors[2]));
      pixels.show(); delay (PERIOD);
    }
  }
}
```

- Compiler messages for ESP32-S2-WROVER-DevKit-Lipo-USB

```
Sketch uses 291526 bytes (22%) of program storage space.
Maximum is 1310720 bytes.
Global variables use 27596 bytes (8%) of dynamic memory,
leaving 300084 bytes for local variables. Maximum is
327680 bytes.
```

- After running sketch on ESP32-S2-WROVER-DevKit-Lipo-USB composite device will be installed with TinyUSB DFU_RT, CDC and ESP32-S2 Firmware MSC devices.
- In terminal connected to USB-Serial CH340 following messages will be sent from ESP32-S2-DevKit-Lipo:

System messages

```
ESP-ROM:esp32s2-rc4-20191025
Build:Oct 25 2019
rst:0x1 (POWERON),boot:0x8 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3ffe6100,len:0x524
load:0x4004c000,len:0xa70
load:0x40050000,len:0x2958
entry 0x4004c18c
```

Sent from setup section

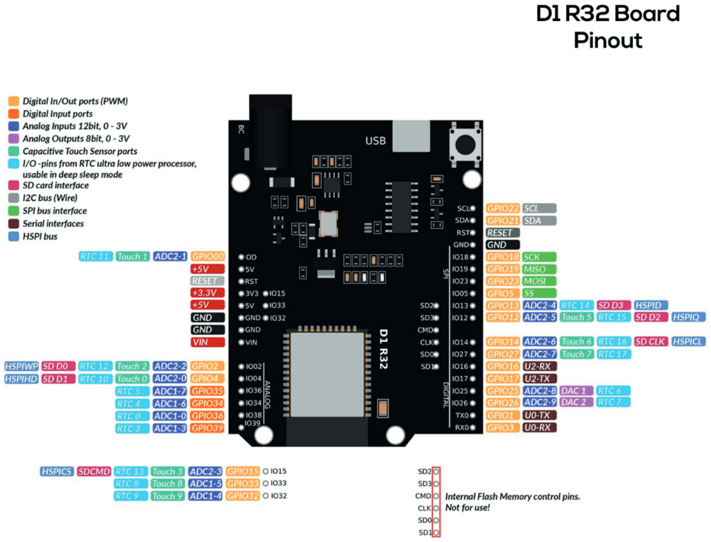
```
Hello World!
```

Sent from loop2 task and will count every 2 sec

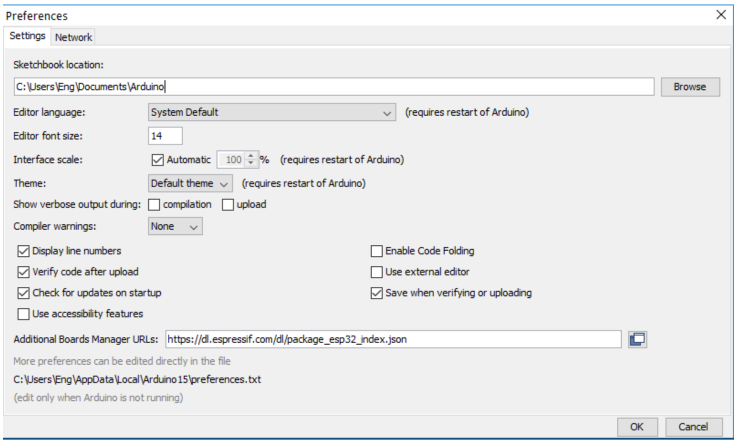
```
Hello World - 0
Hello World - 1
```

Arduino D1 R32 ESP32 board

- For using ESP32 boards like D1 R32


D1 R32 Board Pinout
Digital In/Out ports (PWM)
Digital Input ports
Analog Inputs 12bit, 0 - 3V
Analog Outputs 8bit, 0 - 3V
Capacitive Touch Sensor ports
I/O -pins from RTC ultra low power processor, usable in deep sleep mode
SD card interface
I2C bus (Wire)
SPI bus interface
Serial interfaces
HSPI bus
USB
D1 R32
SPI
ANALOG
DIGITAL
RTC 11 Touch 1 ADC2-1 GPIO00 OD
+5V 5V
RESET RST
+3.3V 3V3 IO15
+5V 5V IO33
GND GND IO32
GND GND
VIN VIN
HSPIWP SD D0 RTC 12 Touch 2 ADC2-2 GPIO2 IO02
HSPIHD SD D1 RTC 10 Touch 0 ADC2-0 GPIO4 IO04
RTC 5 ADC1-7 GPIO35 IO36
RTC 4 ADC1-6 GPIO34 IO34
RTC 0 ADC1-0 GPIO36 IO38
RTC 3 ADC1-3 GPIO39 IO39
SCL GPIO22 SCL
SDA GPIO21 SDA
RST RESET
GND GND
IO18 GPIO18 SCK
IO19 GPIO19 MISO
IO23 GPIO23 MOSI
IO05 GPIO5 SS
SD2 IO13 GPIO13 ADC2-4 RTC 14 SD D3 HSPID
SD3 IO12 GPIO12 ADC2-5 Touch 5 RTC 15 SD D2 HSPIQ
CMD
CLK IO14 GPIO14 ADC2-6 Touch 6 RTC 16 SD CLK HSPICL
SD0 IO27 GPIO27 ADC2-7 Touch 7 RTC 17
SD1 IO16 GPIO16 U2-RX
IO17 GPIO17 U2-TX
IO25 GPIO25 ADC2-8 DAC 1 RTC 6
IO26 GPIO26 ADC2-9 DAC 2 RTC 7
TX0 GPIO1 U0-TX
RX0 GPIO3 U0-RX
HSPICS SDCMD RTC 13 Touch 3 ADC2-3 GPIO15 IO15
RTC 8 Touch 8 ADC1-5 GPIO33 IO33
RTC 9 Touch 9 ADC1-4 GPIO32 IO32
SD2 SD3 CMD CLK SD0 SD1
Internal Flash Memory control pins. Not for use!

- In Preferences add URL: https://dl.espressif.com/dl/package_esp32_index.json



Sketchbook location: C:\Users\Eng\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 14

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: □ compilation □ upload

Compiler warnings: None

☑ Display line numbers □ Enable Code Folding

☑ Verify code after upload □ Use external editor

☑ Check for updates on startup ☑ Save when verifying or uploading

□ Use accessibility features

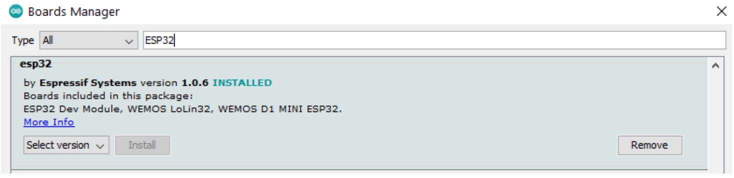
Additional Boards Manager URLs: https://dl.espressif.com/dl/package_esp32_index.json

More preferences can be edited directly in the file

C:\Users\Eng\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

- Install esp32 in board manager

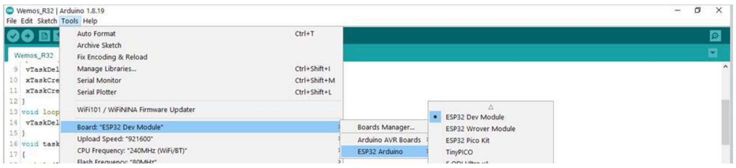


Type All ESP32

esp32

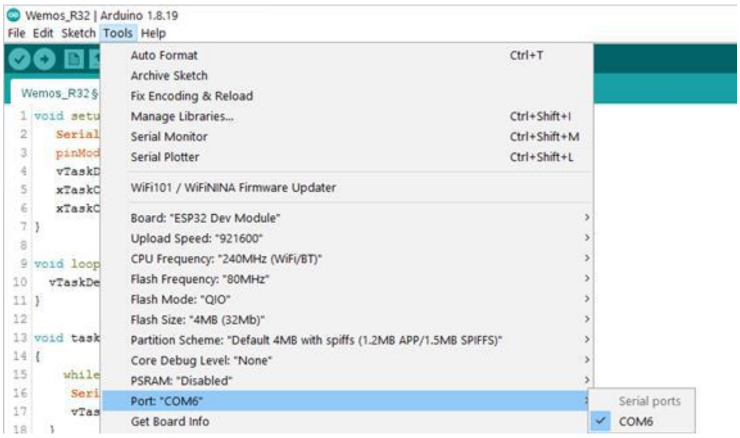
by **Espressif Systems** version **1.0.6** INSTALLED
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

- Connect the board to Windows PC
- Install CH340 USB serial driver if needed and verify the port in "Device Manager": COM6 for example
- Install "ESP32 Dev Module" in board manager


Wemos_R32 | Arduino 1.8.19
File Edit Sketch Tools Help
Wemos_R32
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Boards Manager...
Arduino AVR Boards
ESP32 Arduino
ESP32 Dev Module
ESP32 Wrover Module
ESP32 Pico Kit
TinyPICO

Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"

- Setup USB serial port as verified above


Wemos_R32 | Arduino 1.8.19
File Edit Sketch Tools Help
Wemos_R32 §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Serial ports
COM6

Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Port: "COM6"

Multitasking version of "Hello World & Blinking LED" test for ESP32

- Create new project "HelloWorld" and put the sketch:

```
void setup() {
  Serial.begin(112500);
  // By default the LED is connected to IO02
  pinMode(2, OUTPUT);
  // This will print default SPI pins
  Serial.println("Default SPI pins:");
  Serial.print("MOSI: "); Serial.println(MOSI);
  Serial.print("MISO: "); Serial.println(MISO);
  Serial.print("SCK: ");  Serial.println(SCK);
  Serial.print("SS: ");   Serial.println(SS);
  vTaskDelay(1000 / portTICK_PERIOD_MS);
  xTaskCreate(task1,"task1", 2048, NULL,1,NULL);
  xTaskCreate(task2,"task2", 2048, NULL,1,NULL);
}
void loop() {
  vTaskDelay(1000 / portTICK_PERIOD_MS);
}
void task1( void * parameter ) {
  while(1) {
    Serial.println("Hello World!");
    vTaskDelay(2000 / portTICK_PERIOD_MS);
  }
}
void task2( void * parameter) {
  while(1) {
    digitalWrite(2, HIGH);
    vTaskDelay(100 / portTICK_PERIOD_MS);
    digitalWrite(2, LOW);
    vTaskDelay(100 / portTICK_PERIOD_MS);
  }
}
```

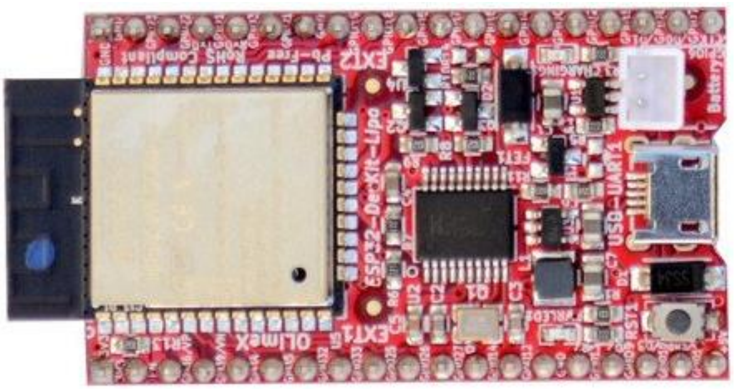
- After compilation will see:

```
Sketch uses 204926 bytes (15%) of program storage space.
Maximum is 1310720 bytes.
Global variables use 13416 bytes (4%) of dynamic memory,
leaving 314264 bytes for local variables. Maximum is
327680 bytes.
```

- After uploading sketch will see fast blinking LED and following messages in terminal to USB serial port:

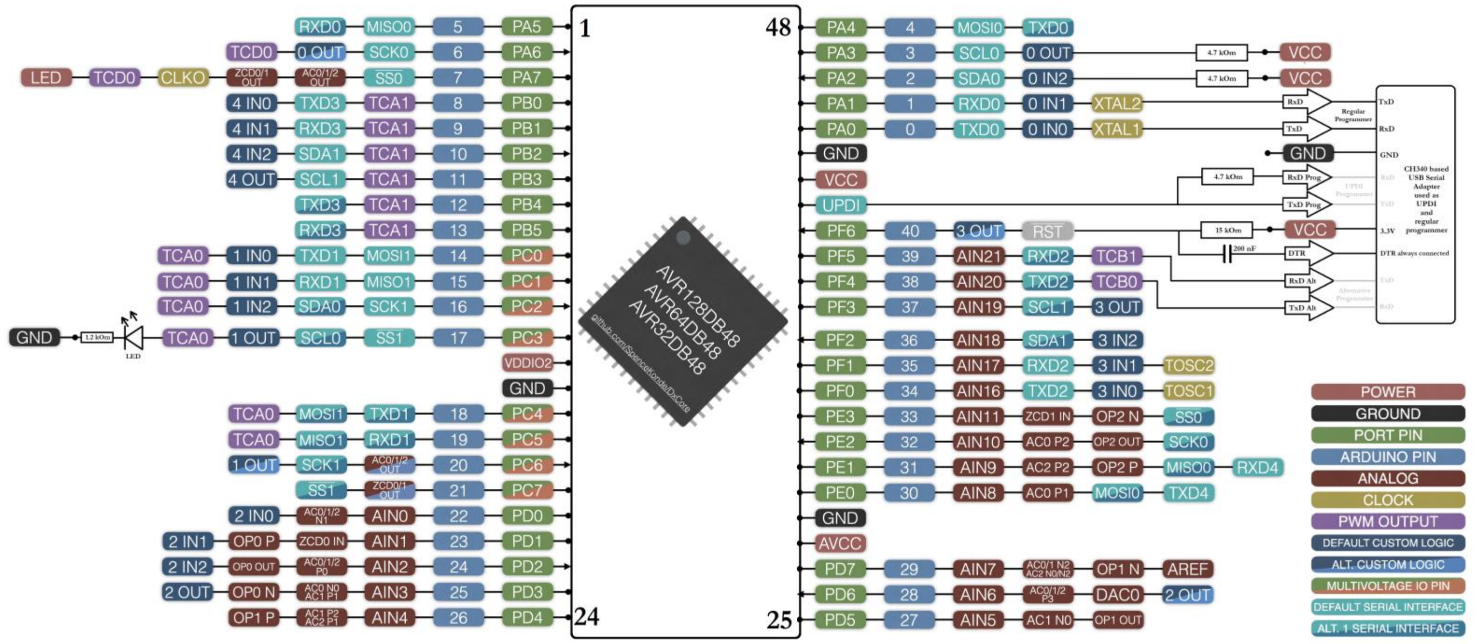
```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x
00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:10944
load:0x40080400,len:6388
entry 0x400806b4
Default SPI pins:        Default settings belongs to VSPI
MOSI: 23
MISO: 19
SCK: 18
SS: 5
Hello World!                 Will be repeated every 2 sec
Hello World!
```

Alternative ESP32 development boards from Olimex based on ESP32-WROOM-32 or ESP32-WROVER WiFi/BLE modules



Variants compatible with Arduino D1 R32 ESP32 board: ESP32-DevKit-Lipo and ESP32-DevKit-Lipo-EA

AVR128db48 Arduino boards

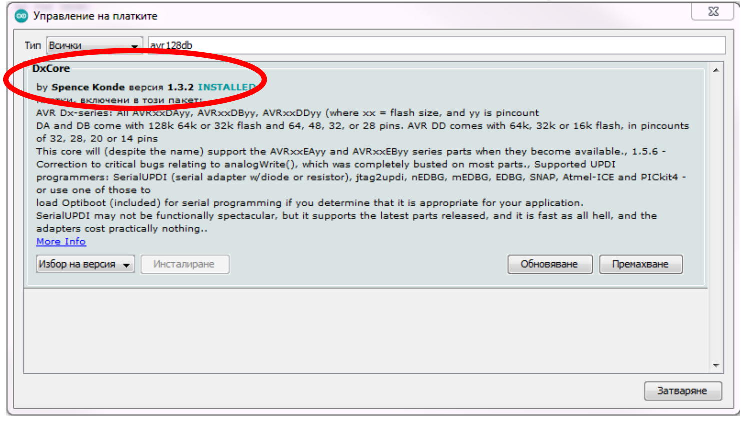

RXD0 MISO0 5 PA5
TCD0 0 OUT SCK0 6 PA6
LED TCD0 CLKO ZCD0/1 OUT AC0/1/2 OUT SS0 7 PA7
4 IN0 TXD3 TCA1 8 PB0
4 IN1 RXD3 TCA1 9 PB1
4 IN2 SDA1 TCA1 10 PB2
4 OUT SCL1 TCA1 11 PB3
TXD3 TCA1 12 PB4
RXD3 TCA1 13 PB5
TCA0 1 IN0 TXD1 MOSI1 14 PC0
TCA0 1 IN1 RXD1 MISO1 15 PC1
TCA0 1 IN2 SDA0 SCK1 16 PC2
GND 1.2 kOm LED TCA0 1 OUT SCL0 SS1 17 PC3
VDDIO2
GND
TCA0 MOSI1 TXD1 18 PC4
TCA0 MISO1 RXD1 19 PC5
1 OUT SCK1 AC0/1/2 OUT 20 PC6
SS1 ZCD0/1 OUT 21 PC7
2 IN0 AC0/1/2 N1 AIN0 22 PD0
2 IN1 OP0 P ZCD0 IN AIN1 23 PD1
2 IN2 OP0 OUT AC0/1/2 P0 AIN2 24 PD2
2 OUT OP0 N AC0 N0 AC1 P1 AIN3 25 PD3
OP1 P AC1 P2 AC2 P1 AIN4 26 PD4
1
24
48
25
AVR128DB48
AVR64DB48
AVR32DB48
github.com/SpenceKonde/DxCore
PA4 4 MOSI0 TXD0
PA3 3 SCL0 0 OUT 4.7 kOm VCC
PA2 2 SDA0 0 IN2 4.7 kOm VCC
PA1 1 RXD0 0 IN1 XTAL2 RxD TxD
PA0 0 TXD0 0 IN0 XTAL1 TxD RxD
Regular Programmer
GND GND GND
VCC 4.7 kOm RxD Prog RxD
UPDI TxD Prog TxD
UPDI Programmer
PF6 40 3 OUT RST 15 kOm VCC 3.3V
200 nF DTR DTR always connected
PF5 39 AIN21 RXD2 TCB1 RxD Alt TxD
PF4 38 AIN20 TXD2 TCB0 TxD Alt RxD
Alternative Programmer
CH340 based USB Serial Adapter used as UPDI and regular programmer
PF3 37 AIN19 SCL1 3 OUT
PF2 36 AIN18 SDA1 3 IN2
PF1 35 AIN17 RXD2 3 IN1 TOSC2
PF0 34 AIN16 TXD2 3 IN0 TOSC1
PE3 33 AIN11 ZCD1 IN OP2 N SS0
PE2 32 AIN10 AC0 P2 OP2 OUT SCK0
PE1 31 AIN9 AC2 P2 OP2 P MISO0 RXD4
PE0 30 AIN8 AC0 P1 MOSI0 TXD4
GND
AVCC
PD7 29 AIN7 AC0/1 N2 AC2 N0/N2 OP1 N AREF
PD6 28 AIN6 AC0/1/2 P3 DAC0 2 OUT
PD5 27 AIN5 AC1 N0 OP1 OUT
POWER
GROUND
PORT PIN
ARDUINO PIN
ANALOG
CLOCK
PWM OUTPUT
DEFAULT CUSTOM LOGIC
ALT. CUSTOM LOGIC
MULTIVOLTAGE IO PIN
DEFAULT SERIAL INTERFACE
ALT. 1 SERIAL INTERFACE

Self-made AVR128db48 Arduino like board based on QFP adapter board

Other boards notes:

- Arduino UNO – install windows driver for USB-Serial CH340 adapter,
- Olimexino Nano – install windows driver for Arduino Leonardo compatible boards,
- Set in Tools → Board → Arduino AVR Boards → Arduino UNO or Arduino Leonardo respectively,
- Set in Tools → Port → corresponding COM port,
- LED pin may be different for different boards – change it in “Blinking LED” test sketch.

- For using AVR128DB48 boards from Anton do:
- Add URL in Preferences: `http://drazzy.com/package_drazzy.com_index.json`
- Install DxCore ver. 1.3.2 in board manager


Управление на платките

Тип Всички avr128db

DxCore

by **Spence Konde** версия **1.3.2** INSTALLED
Платки, включени в този пакет:
AVR Dx-series: All AVRxxDAyy, AVRxxDByy, AVRxxDDyy (where xx = flash size, and yy is pincount
DA and DB come with 128k 64k or 32k flash and 64, 48, 32, or 28 pins. AVR DD comes with 64k, 32k or 16k flash, in pincounts of 32, 28, 20 or 14 pins
This core will (despite the name) support the AVRxxEAyy and AVRxxEByy series parts when they become available., 1.5.6 - Correction to critical bugs relating to analogWrite(), which was completely busted on most parts., Supported UPDI programmers: SerialUPDI (serial adapter w/diode or resistor), jtag2updi, nEDBG, mEDBG, EDBG, SNAP, Atmel-ICE and PICkit4 - or use one of those to
load Optiboot (included) for serial programming if you determine that it is appropriate for your application.
SerialUPDI may not be functionally spectacular, but it supports the latest parts released, and it is fast as all hell, and the adapters cost practically nothing..
More Info

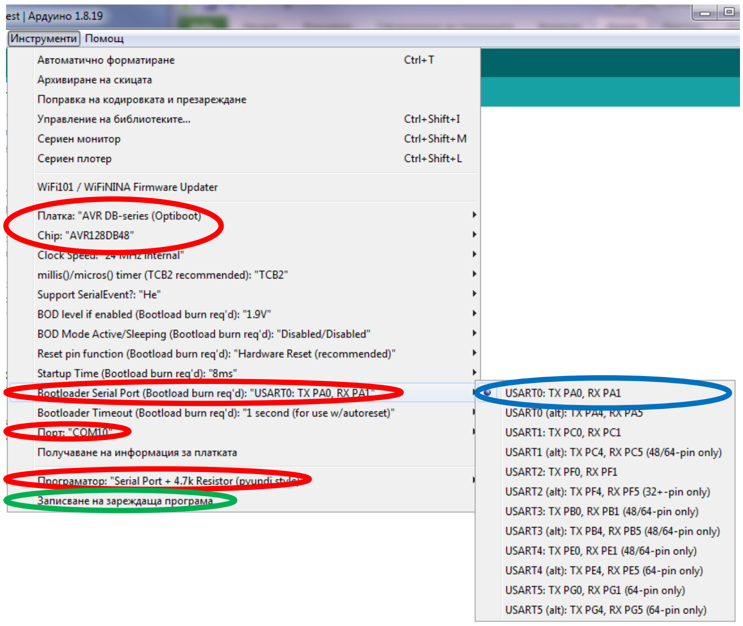
- Connect CP2102 USB to UART Bridge to Windows PC
- Install CP2102 USB driver if needed and verify the port: COM10 for example

UPDI programmer (to burn bootloader)

- Connect CP2102 USB to UART Bridge to the board
  - Rx ← 4.7k res. → Tx → AVR128DB48 UPDI (pin 41),
  - DTR → 200nF → RST (p. 40), GND, VCC (3.3V)
- Programmer: "Serial Port + 4.7k Resistor (pyupdi style)"
- Usage: Tools → Burn Bootloader
- Usage: Sketch → Upload Using Programmer

Regular serial programmer

- Connect CP2102 USB to UART Bridge to the board
  - CP2102/TTL-232R Tx → AVR128DB48 Rx0 (p. 45)
  - CP2102/TTL-232R Rx ← AVR128DB48 Tx0 (p. 44)
  - DTR → 200nF → RST (p. 40), GND, VCC (3.3V)
- Usage: Sketch → Upload



Автоматично форматиране Ctrl+T
Архивиране на скицата
Поправка на кодировката и презареждане
Управление на библиотеките... Ctrl+Shift+I
Сериен монитор Ctrl+Shift+M
Сериен плотер Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Платка: "AVR DB-series (Optiboot)"
Chip: "AVR128DB48"
Clock Speed: "24 MHz internal"
millis()/micros() timer (TCB2 recommended): "TCB2"
Support SerialEvent?: "He"
BOD level if enabled (Bootload burn req'd): "1.9V"
BOD Mode Active/Sleeping (Bootload burn req'd): "Disabled/Disabled"
Reset pin function (Bootload burn req'd): "Hardware Reset (recommended)"
Startup Time (Bootload burn req'd): "8ms"
Bootloader Serial Port (Bootload burn req'd): "USART0: TX PA0, RX PA1"
Bootloader Timeout (Bootload burn req'd): "1 second (for use w/autoreset)"
Порт: "COM10"
Получаване на информация за платката
Програматор: "Serial Port + 4.7k Resistor (pyupdi style)"
Записване на зареждаща програма

● USART0: TX PA0, RX PA1
USART0 (alt): TX PA4, RX PA5
USART1: TX PC0, RX PC1
USART1 (alt): TX PC4, RX PC5 (48/64-pin only)
USART2: TX PF0, RX PF1
USART2 (alt): TX PF4, RX PF5 (32+-pin only)
USART3: TX PB0, RX PB1 (48/64-pin only)
USART3 (alt): TX PB4, RX PB5 (48/64-pin only)
USART4: TX PE0, RX PE1 (48/64-pin only)
USART4 (alt): TX PE4, RX PE5 (64-pin only)
USART5: TX PG0, RX PG1 (64-pin only)
USART5 (alt): TX PG4, RX PG5 (64-pin only)

Bootloader serial port could be USART2 (alt), but using USART0 the PC COM port for both programming and serial communication with the sketch will be the same.

“Blinking LED” test for avr128db48

```
void setup() {
  // PIN_PC3 for avr128db48
  // may be different for other boards!
  pinMode(17, OUTPUT);
}
void loop() {
  digitalWrite(17, 1);
  delay(100);
  digitalWrite(17, 0);
  delay(100);
}
```