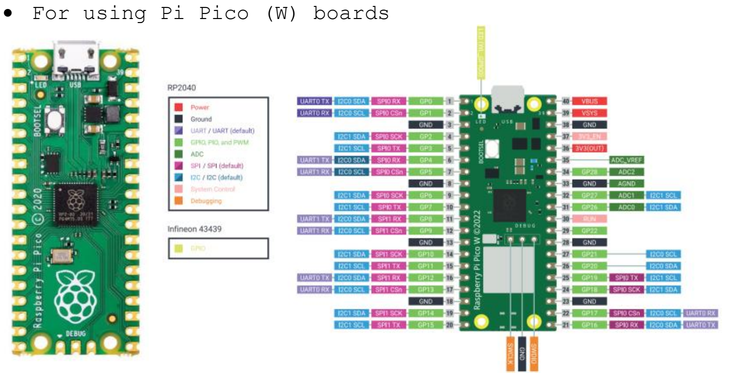
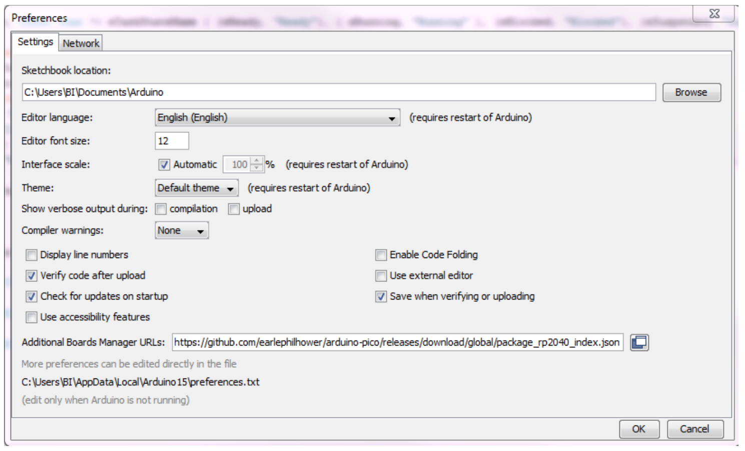
Arduino Pi Pico (W) boards

- 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

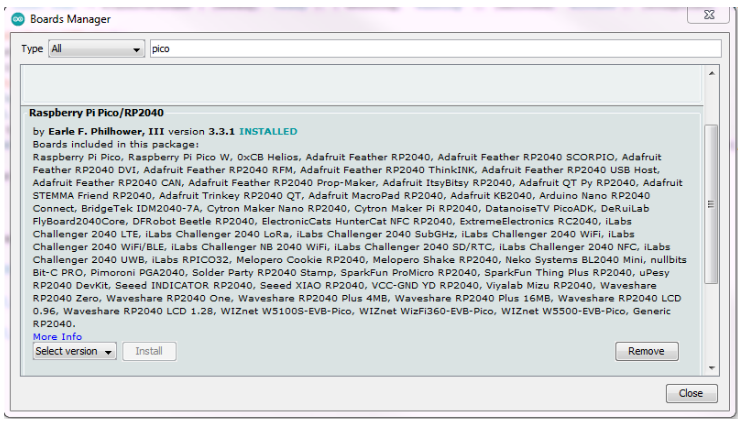
[ ] Display line numbers
[✓] Verify code after upload
[✓] Check for updates on startup
[ ] Use accessibility features
[ ] Enable Code Folding
[ ] Use external editor
[✓] Save when verifying or uploading

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

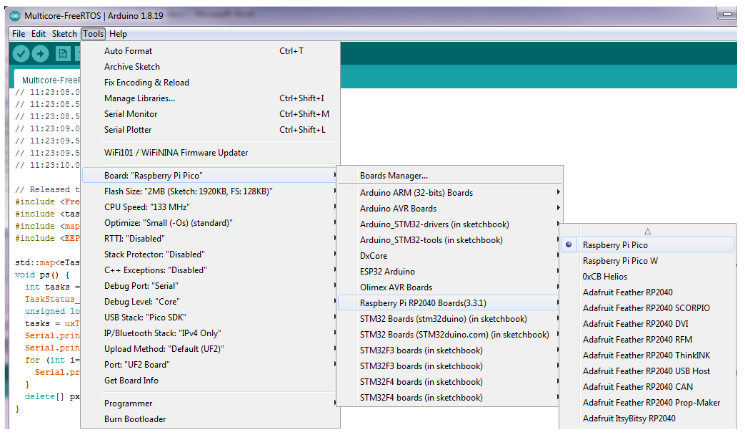
Type All | pico

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

Tools menu: 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; 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"; Get Board Info; Programmer; Burn Bootloader

Board submenu: 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 RP2040 Boards submenu: ● 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

- 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:

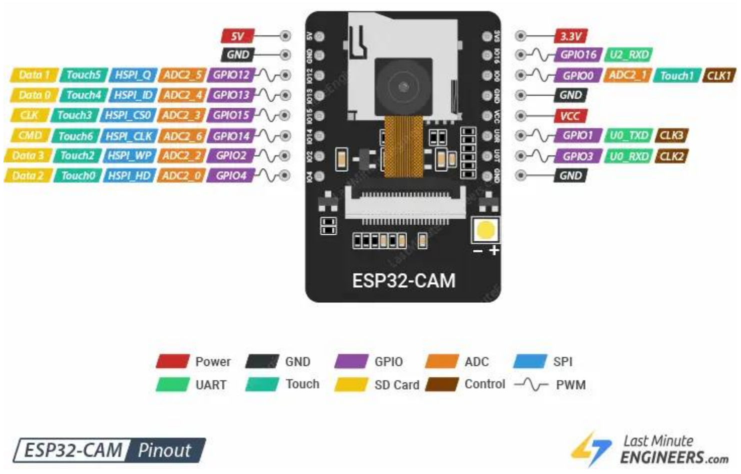
```
#include <FreeRTOS.h>
#include <task.h>
#include <map>
#include <EEPROM.h>
std::map<eTaskState, const char *> eTaskStateName {
{eReady, "Ready"}, { eRunning, "Running" }, {eBlocked,
"Blocked"}, {eSuspended, "Suspended"}, {eDeleted,
"Deleted"} };
void ps() {
  int tasks = uxTaskGetNumberOfTasks();
  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:

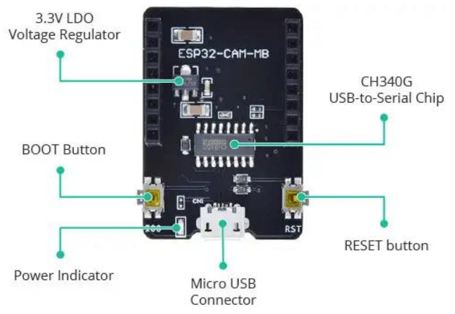
```
C1: Stay on target...
C0: Blue leader standing by...
# Tasks: 9
ID, NAME, STATE, PRIO, CYCLES
0: CORE0             Running    4 191473164
1: IDLE1             Running    0 3622023404
2: IDLE0             Ready      0 3371562651
3: BLINK             Blocked    1 5381238
4: CORE1             Blocked    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
```

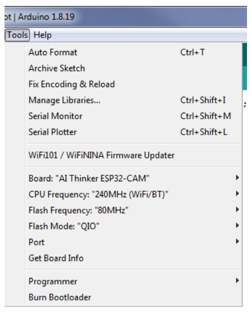
ESP32-CAM

- 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




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
Port
Get Board Info
Programmer
Burn Bootloader

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

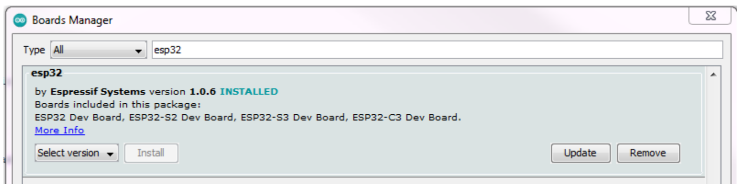
Wifi Camera Robot Car

- 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.

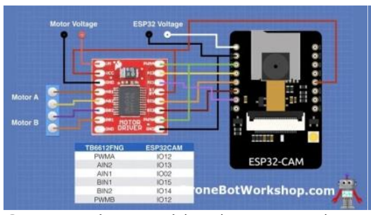
- To compile it use ESP32 Espressif System 1.0.6



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

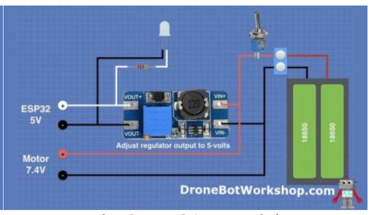
- Follow instructions in about the hardware : https://dronebotworkshop.com/esp32cam-robot-car/

- Main electrical parts



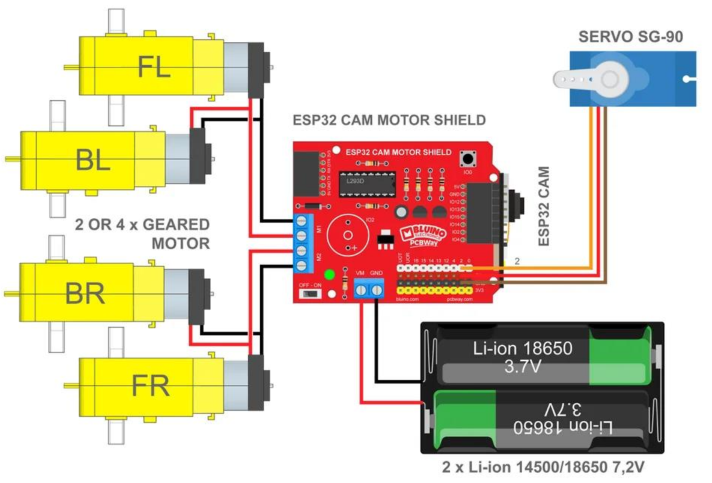
TB6612FNG	ESP32CAM
PWMA	IO12
AIN2	IO13
AIN1	IO02
BIN1	IO15
BIN2	IO14
PWMB	IO12

Camera and motor driver interconnection



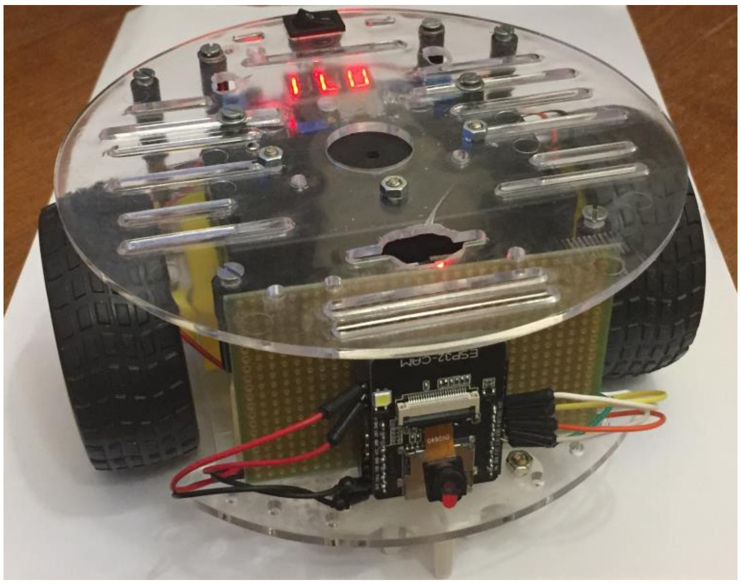
Motor and ESP32-CAM module power

- Power schematics



- Final results

Wifi Camera Robot Car – own implementation



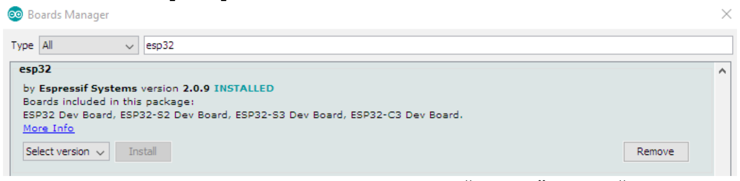
ESP32-S2 board on Arduino IDE

- For using ESP32-S2 boards like:

ESP32-S2-DevKit-Lipo

ESP32-S2-WROVER-DevKit-Lipo-USB

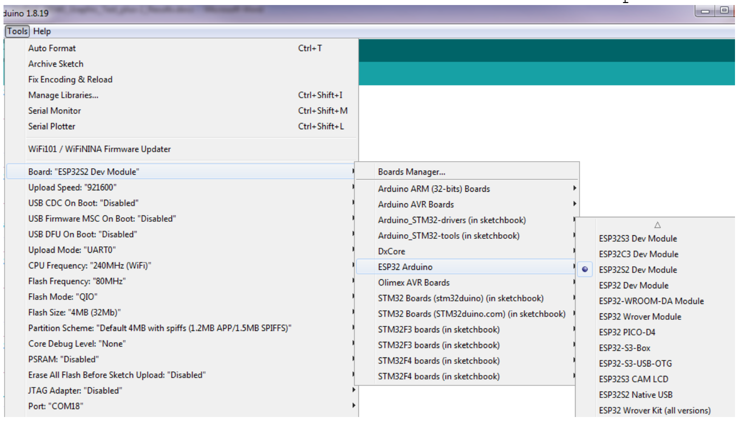
- Install the ESP32-S2 support for Arduino IDE
- In "File" → "Preferences" add URL: https://espressif.github.io/arduino-esp32/package_esp32_index.json
- In "Tools" → "Boards" → "Board Manager" search for the esp32 platform and install ver. 2.0.0 or later


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

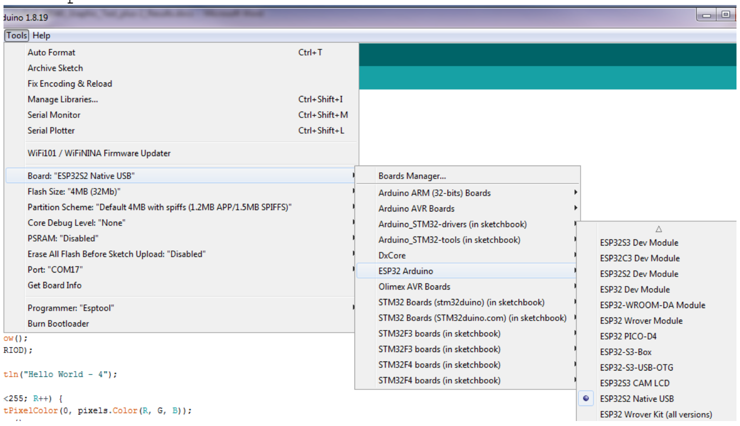
- Restart IDE and select board in "Tools" → "Board:
  - o "ESP32S2 Dev Module" for ESP32-S2-DevKit-Lipo


duino 1.8.19

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)"
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"
Erase All Flash Before Sketch Upload: "Disabled"
JTAG Adapter: "Disabled"
Port: "COM18"

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
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
ESP32 Wrover Kit (all versions)

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


duino 1.8.19

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"

```
ow();
RIOD);

tln("Hello World - 4");

<255; R++) {
tPixelColor(0, pixels.Color(R, G, B));
```

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
ESP32 Wrover Kit (all versions)

- Connect ESP32-S2-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

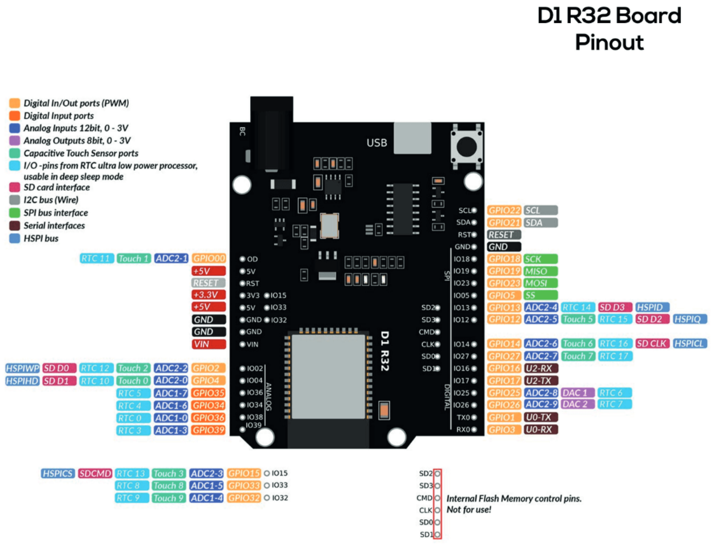
```
Sketch uses 291526 bytes (22%) of program storage space.
Maximum is 1310720 bytes.
Global variables use 27596 bytes (8%) of dynamic memory,
leaving 300084 bytes for local variables. Maximum is
327680 bytes.
```

- 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:

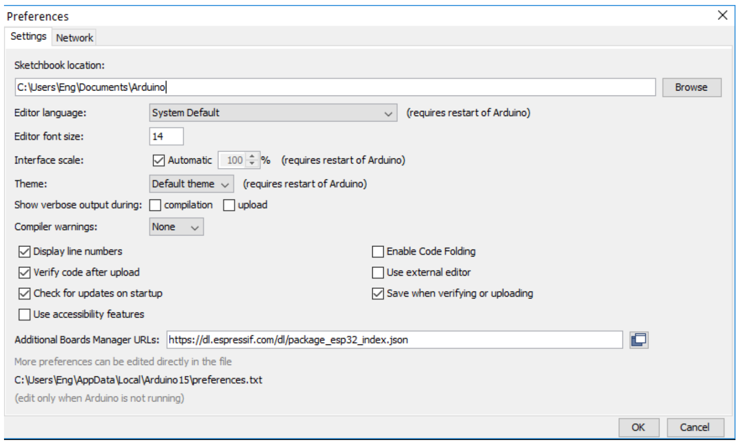
```
ESP-ROM:esp32s2-rc4-20191025                    System messages
Build:Oct 25 2019
rst:0x1 (POWERON),boot:0x8 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3ffe6100,len:0x524
load:0x4004c000,len:0xa70
load:0x40050000,len:0x2958
entry 0x4004c18c
Hello World!                              Sent from setup section
Hello World - 0     Sent from loop2 task and will count every 2 sec
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
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
ANALOG
SCL GPIO22 SCL
SDA GPIO21 SDA
RST RESET
GND GND
IO18 GPIO18 SCK
IO19 GPIO19 MISO
IO23 GPIO23 MOSI
IO05 GPIO5 SS
SPI
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
DIGITAL
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


Preferences

Settings | Network

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

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

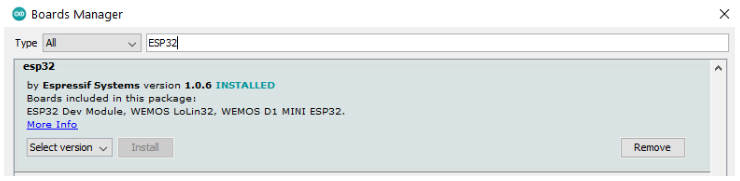
More preferences can be edited directly in the file

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

(edit only when Arduino is not running)

[OK] [Cancel]

- Install esp32 in board manager

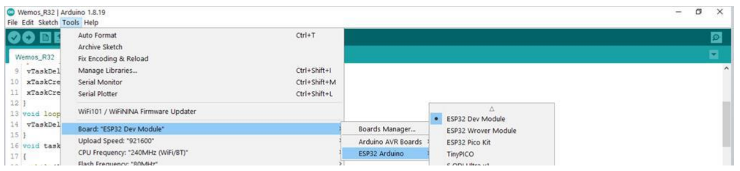

Boards Manager

Type All ESP32

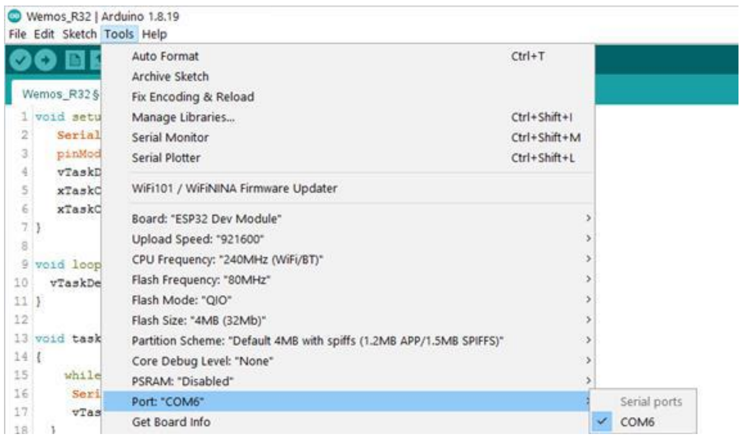
esp32
by **Espressif Systems** version **1.0.6** **INSTALLED**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

[Select version] [Install] [Remove]

- 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
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
Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Boards Manager...
Arduino AVR Boards
ESP32 Arduino
ESP32 Dev Module
ESP32 Wrover Module
ESP32 Pico Kit
TinyPICO

- Setup USB serial port as verified above


Wemos_R32 | Arduino 1.8.19
File Edit Sketch 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
Serial ports

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"
✓ 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:

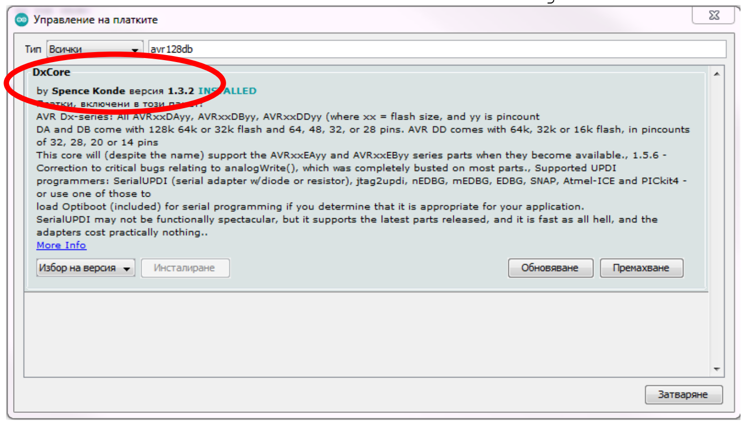
```
Sketch uses 204926 bytes (15%) of program storage space.
Maximum is 1310720 bytes.
Global variables use 13416 bytes (4%) of dynamic memory,
leaving 314264 bytes for local variables. Maximum is
327680 bytes.
```

- 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!
```

AVR128db48 Arduino boards

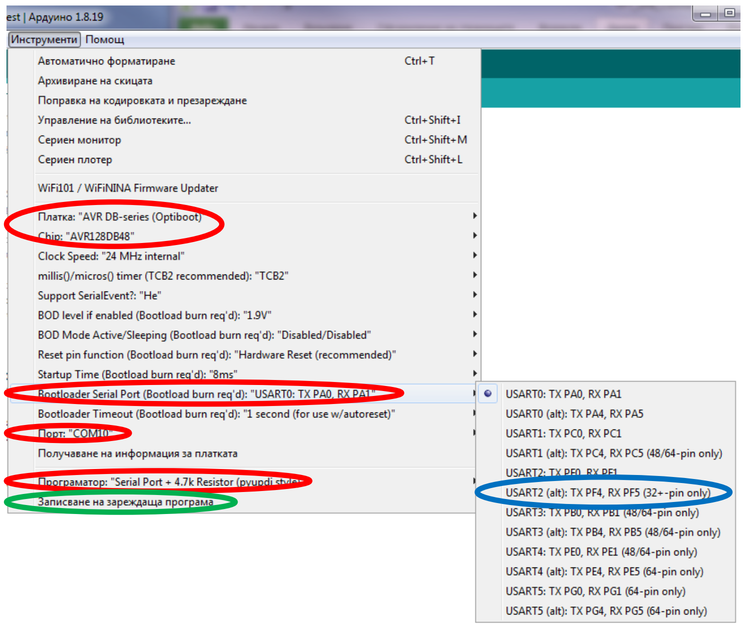
- For using AVR128DB48 boards from Anton do:
- Add URL in Preferences: http://drazzy.com/package_drazzy.com_index.json
- Install DxCore ver. 1.3.2 in board manager



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



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)

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

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.

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

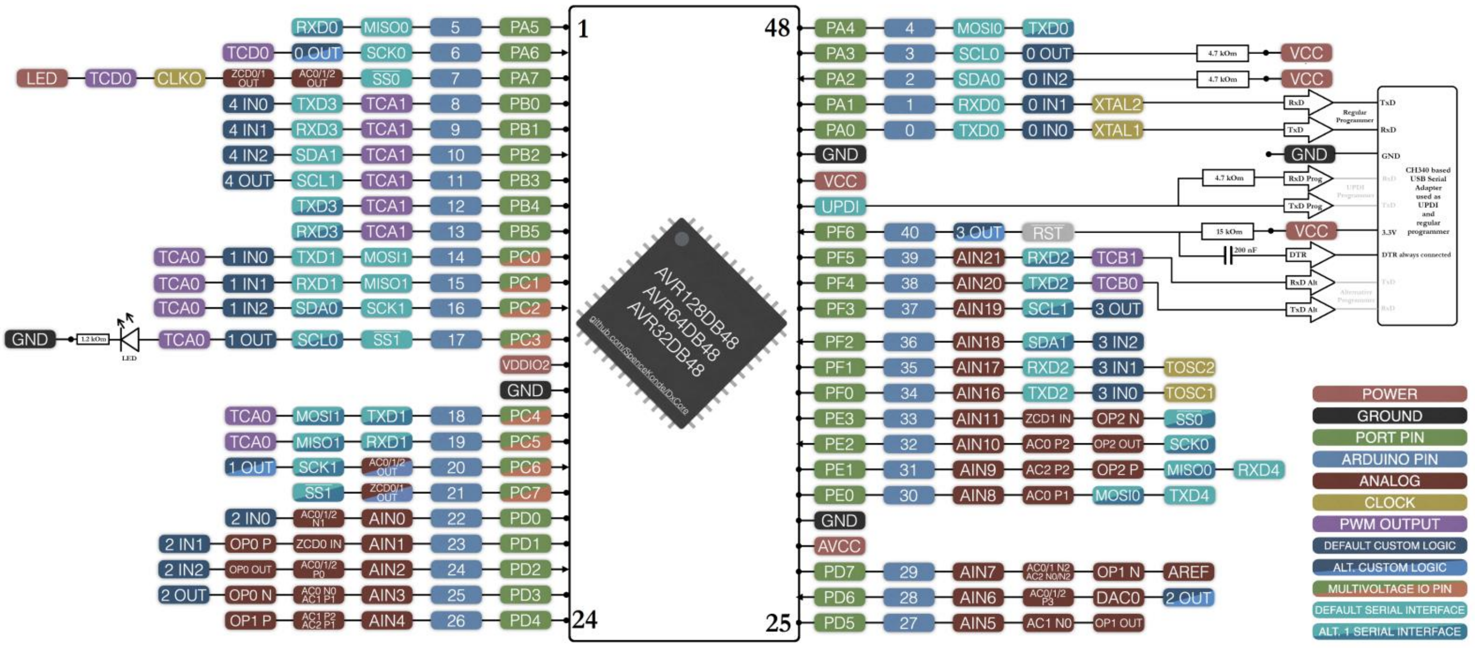
Regular serial programmer

- 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

"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);
}
```

AVR128DB48 on 48 pin QFP adapter board



1
48
24
25
AVR128DB48
AVR64DB48
AVR32DB48
github.com/SpenceKonde/DxCore
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 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
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
PA0 0 TXD0 0 IN0 XTAL1
GND
VCC
UPDI
PF6 40 3 OUT RST
PF5 39 AIN21 RXD2 TCB1
PF4 38 AIN20 TXD2 TCB0
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
TxD RxD Regular Programmer
RxD Prog TxD Prog UPDI Programmer
DTR RxD Alt TxD Alt Alternative Programmer
4.7 kOm
15 kOm
200 nF
TxD RxD GND 3.3V
CH340 based USB Serial Adapter used as UPDI and regular programmer
DTR always connected
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

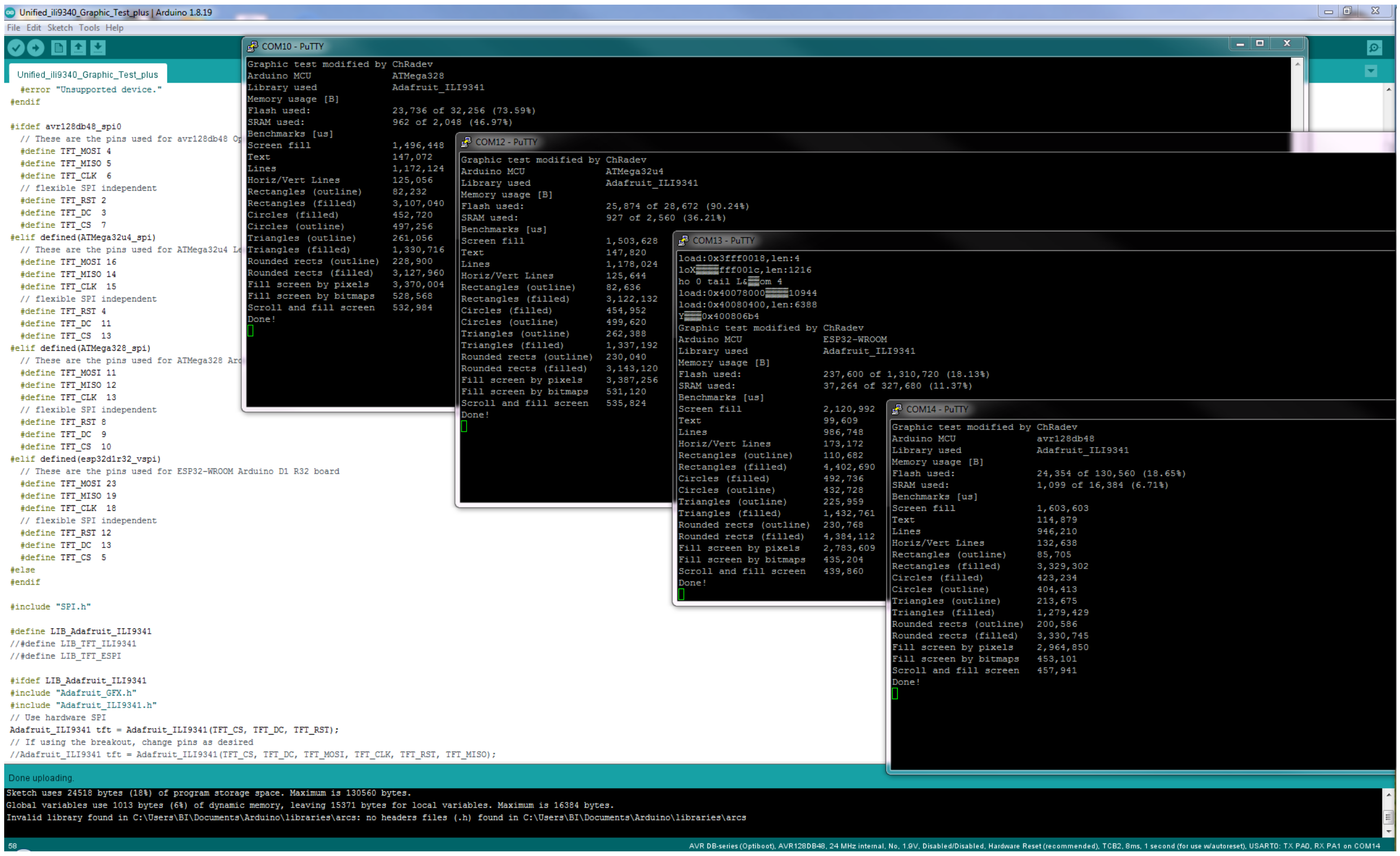
Other boards notes:

- 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.

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 & CYW43439	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 (OUT)	3.3V power input (do not connect to 5V)
2	GND	GND	GND	GND	GND	GND	GND
3	CS	D10	D13	0,#SS, PA7	IO05	GP17	LCD chip select signal, low level enable
4	RESET	D8	D4	PA2 (0,SDA)	IO12	GP21	LCD reset signal, low level reset
5	DC/RS	D9	D11	PA3 (0,SCL)	IO13	GP20	LCD register / data selection signal, high level: register, low level: data
6	SDI(MOSI)	D11	D16	0,MOSI, PA4	IO23	GP16	SPI bus write data signal
7	SCK	D13	D15	0,SCK, PA6	IO18	GP18	SPI bus clock signal
8	LED	VCC-5V	VCC-5V	VCC-5V	VCC-5V	VCC-5V (VBUS)	Backlight control, high level lighting, if not controlled, connect 5V for always bright
9	SDO(MISO)	D12	D14	0,MISO, PA5	IO19	GP19	SPI bus read data signal, if you do not need to the read function, you cannot connect it

All 3 boards are connected to 3.2”SPI TFT display and running Unified graphic test


Unified_ili9340_Graphic_Test_plus | Arduino 1.8.19

```
  #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[illegible]fff001c,len:1216
ho 0 tail L&[illegible]om 4
load:0x40078000[illegible]10944
load:0x40080400,len:6388
Y[illegible]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!
```

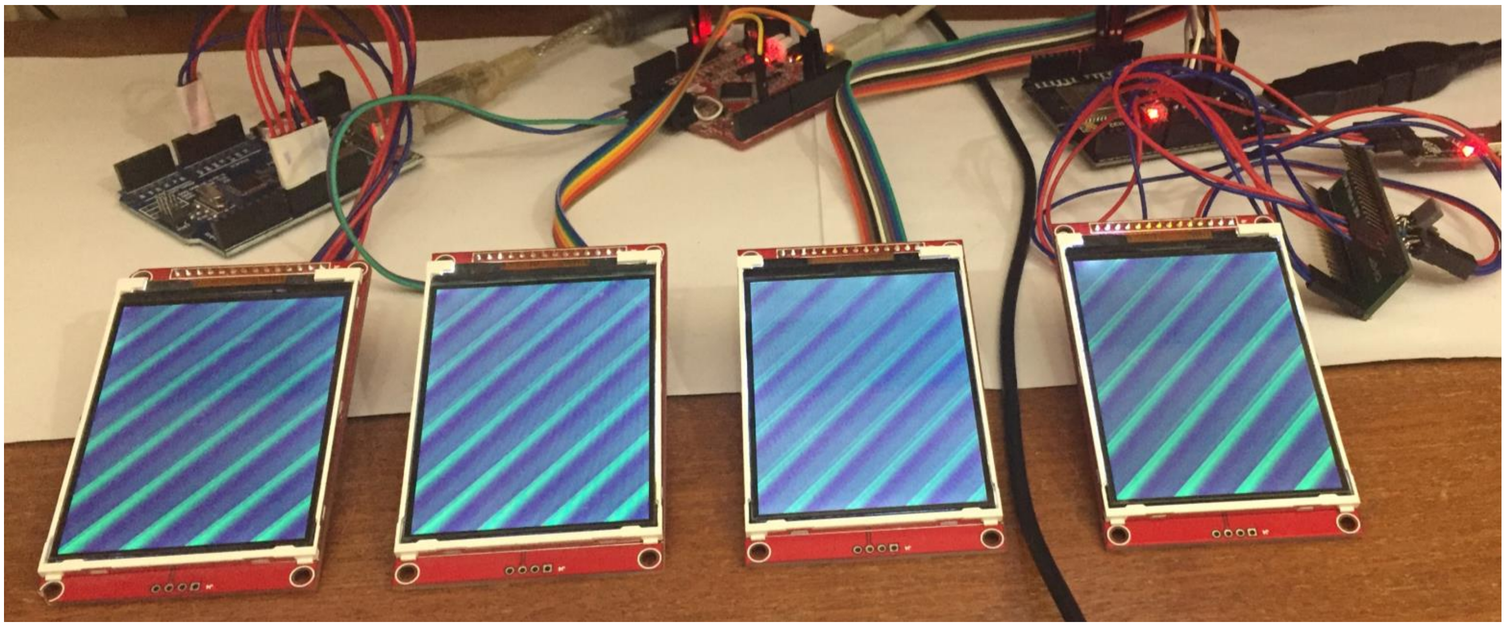
COM14 - PuTTY

```
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
```

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)

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)			AVR128db48
ILI9341 Library used	Adafruit	TFT	Speed up	Adafruit	TFT	Speed up	Adafruit	TFT_eSPI	Speed up	Adafruit	TFT_eSPI	Speed up	Adafruit	TFT_eSPI	Speed up	Adafruit
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%)	269,072 of 1,310,720 (20.53%)		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%)		24,354 of 130,560 (18.65%)
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%)	36,864 of 327,680 (11.25%)		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%)		1,087 of 16,384 (6.63%)
Benchmarks [us]																
Screen fill	1,496,456	870,220	1.720	1,503,900	874,600	1.720	2,120,993	233,151	9.097	604,056	281,577	2.145	497,451	107,972	4.607	1,603,604
Text	147,088	60,416	2.435	147,820	60,724	2.434	99,610	15,346	6.491	45,452	18,831	2.414	30,599	8,085	3.785	114,885
Lines	1,172,116	242,732	4.829	1,178,004	243,988	4.828	986,748	89,909	10.975	454,856	101,897	4.464	304,234	42,741	7.118	946,199
Horiz/Vert Lines	125,064	71,336	1.753	125,656	71,696	1.753	173,171	20,128	8.603	50,042	23,541	2.126	40,853	9,078	4.500	132,637
Rectangles (outline)	82,228	45,844	1.794	82,632	46,076	1.793	110,682	12,727	8.697	32,657	14,932	2.187	26,417	5,773	4.576	85,703
Rectangles (filled)	3,107,060	1,807,436	1.719	3,122,844	1,816,740	1.719	4,402,687	484,050	9.096	1,253,856	584,372	2.146	1,032,576	224,086	4.608	3,329,307
Circles (filled)	452,728	284,064	1.594	454,916	285,536	1.593	492,735	63,960	7.704	167,914	71,149	2.360	126,969	28,025	4.531	423,221
Circles (outline)	497,252	135,580	3.668	499,604	136,148	3.670	432,728	33,343	12.978	199,626	37,258	5.358	133,263	15,561	8.564	404,412
Triangles (outline)	261,056	59,496	4.388	262,392	59,808	4.387	225,959	22,013	10.265	101,400	23,636	4.290	68,473	10,319	6.636	213,681
Triangles (filled)	1,330,720	694,456	1.916	1,337,200	698,032	1.916	1,432,757	164,864	8.691	429,998	195,995	2.194	345,244	75,450	4.576	1,279,412
Rounded rects (outline)	228,892	100,004	2.289	230,024	100,532	2.288	230,767	20,954	11.013	92,280	23,635	3.904	65,233	9,576	6.812	200,582
Rounded rects (filled)	3,127,968	1,976,936	1.582	3,143,588	1,987,180	1.582	4,384,111	487,395	8.995	1,257,871	586,292	2.145	1,032,024	225,027	4.586	3,330,751
Fill screen by pixels	3,369,992	918,732	3.668	3,387,308	923,492	3.668	2,783,609	835,657	3.331	1,255,234	504,753	2.487	805,373	229,258	3.513	2,964,859
Fill screen by bitmaps	528,576	855,088	0.618	531,112	859,520	0.618	435,203	840,458	0.518	66,438	520,180	0.128	70,363	234,904	0.300	453,099
Scroll and fill screen	**532,988**	**855,696**	**0.623**	**535,808**	**860,132**	**0.623**	**439,860**	**845,475**	**0.520**	**69,357**	**521,011**	**0.133**	**71,933**	**235,385**	**0.306**	**457,946**
Min	82,228	45,844		82,632	46,076		99,610	12,727		32,657	14,932		26,417	5,773		85,703
Avg	**1,097,346**	**598,536**	**1.833**	**1,102,854**	**601,614**	**1.833**	**1,250,108**	**277,962**	**4.497**	**405,402**	**233,937**	**1.733**	**310,067**	**97,416**	**3.183**	**1,062,687**
Max	3,369,992	1,976,936		3,387,308	1,987,180		4,402,687	845,475		1,257,871	586,292		1,032,576	235,385		3,330,751
Sum	16,460,184	8,978,036		16,542,808	9,024,204		18,751,620	4,169,430		6,081,037	3,509,059		4,651,005	1,461,240		15,940,298
DrawWithDMA test (bonsing of 42 colored and numbered circles)								21.6 fps at CPU 240MHz SPI 27MHz			17.8 fps at CPU 133MHz SPI 27MHz			46.5 fps at CPU 250MHz SPI 62.5MHz	2.2 / 2.6	

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;
- Numbers in "Speed up" column means the operation is that many times faster;
- Overclocking includes increasing of SPI and CPU speed up to 62.5MHz and 250MHz respectively and suggested solution by Bodemar in his Github issue 1460 (working reliably even with 30cm long wires).

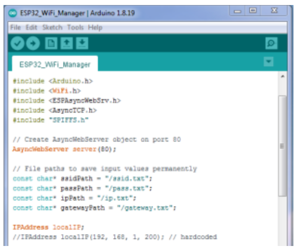
Useful links for display drawing with DMA and speed assessment:

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.

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

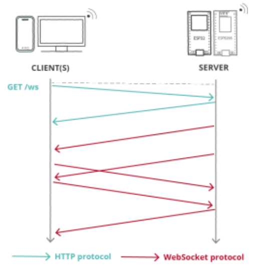
Startup projects working on ESP32 S2 Olimex boards and based on ESPAsyncWebServer library for Arduino:



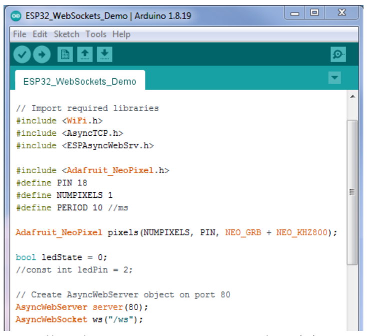
```
#include <Arduino.h>
#include <WiFi.h>
#include <ESPAsyncWebSrv.h>
#include <AsyncTCP.h>
#include "SPIFFS.h"

// 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)

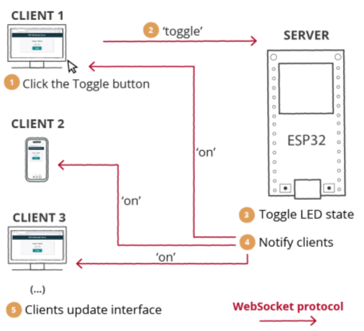
Application uses SPIFS on ESP32 systems to hold web and configuration files which have to be written manually by "ESP32 Sketch Data Upload" tool of Arduino IDE. The application first runs in AP mode asking for connection credentials of the local router. After storing them in FS files and restart it runs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



- ESP32 WebSocket Server: Control Outputs (Arduino IDE)



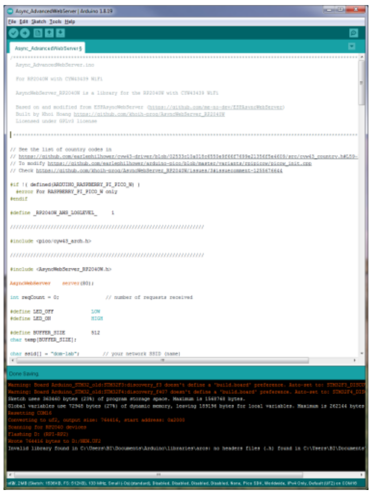
```
// Import required libraries
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebSrv.h>

#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");
```

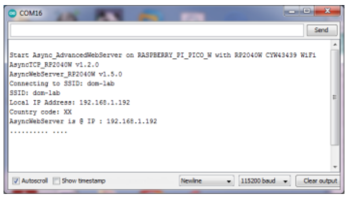


Application on ESP32 runs in STA mode with credentials defined in the sketch and open WebSocket server to control the LED. Its status can be changed by any client and will be updated at all the clients.

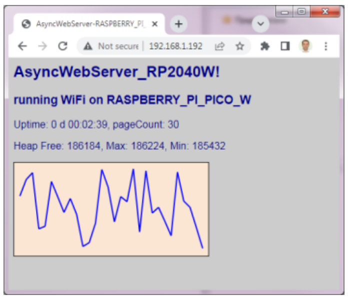
It was used Adafruit NeoPixel library to run above projects on Olimex ESP32 S2 series of boards with RGB instead of regular LED.

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





```
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
.......... ....
```



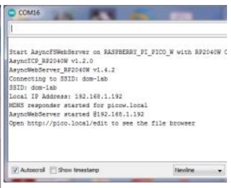
AsyncWebServer_RP2040W!

running WiFi on RASPBERRY_PI_PICO_W

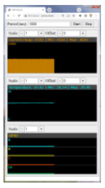
Uptime: 0 d 00:02:39, pageCount: 30

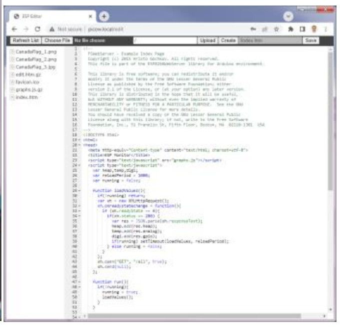
Heap Free: 186184, Max: 186224, Min: 185432

- Async_AdvancedWebServer



```
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
```

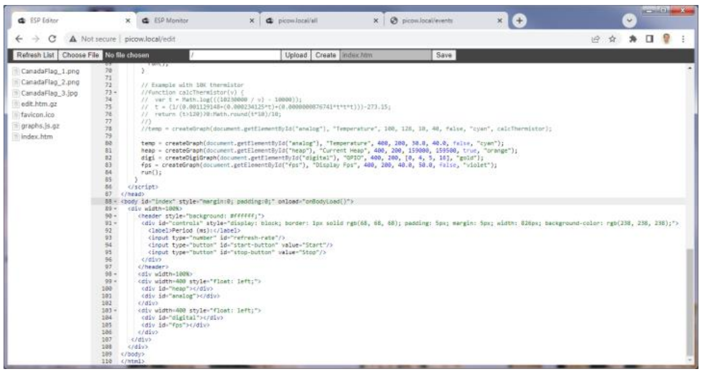




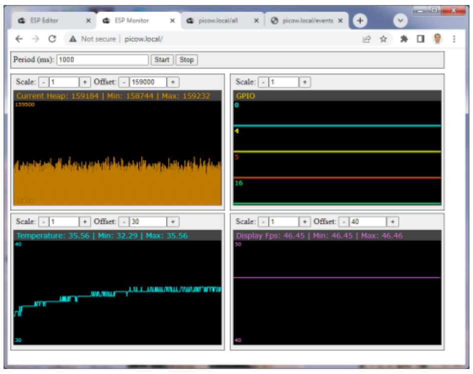
- AsyncFSWebServer (library ver. 1.5.0 did not run out of the box)

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.

AsyncFSWebServer and DrawWithDMA combined multicore application for Raspberry Pi Pico W 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). This makes CPU overclocking useless. 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.



+



Period (ms): 1000 Start Stop

Scale: 1 Offset: 159000 — Current Heap: 159184 | Min: 158744 | Max: 159232

Scale: 1 — GPIO

Scale: 1 Offset: 30 — Temperature: 35.56 | Min: 32.29 | Max: 35.56

Scale: 1 Offset: 40 — Display Fps: 46.45 | Min: 46.45 | Max: 46.46

+

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:

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