

Ustawienie i konfiguracja bibliotek niezbędnych do programowania Waleta

1. Instalacja platformy Arduino

Software dla różnych systemów operacyjnych można znaleźć pod tym linkiem:

<https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE>

Kolejno należy wgrać do Arduino biblioteki procesorów ESP:

Installing ESP32 Add-on in Arduino IDE

1. In your Arduino IDE, go to File> Preferences.
2. Open the Boards Manager. Go to Tools > Board > Boards Manager...
3. Search for ESP32 and press install button for the “ESP32 by Espressif Systems“:
4. That's it. It should be installed after a few seconds.

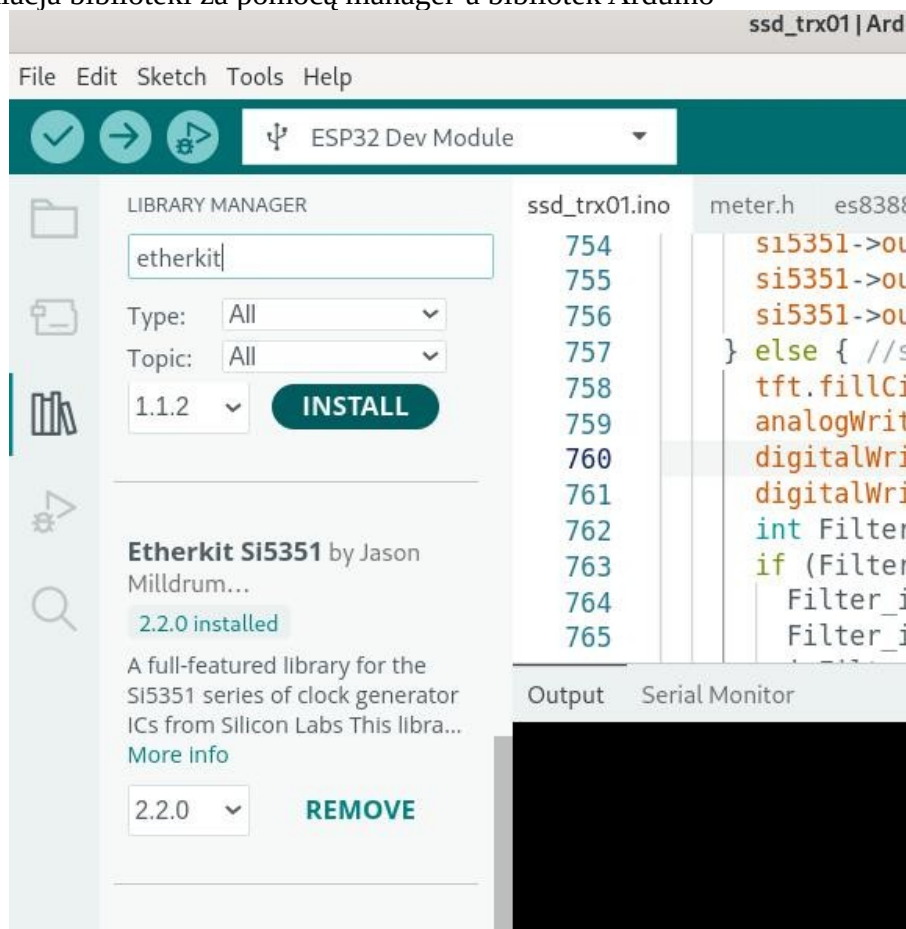
Pomocny film: <https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>

Po dołączeniu konwertera USB/RS do procesora należy wybrać odpowiedni procesor oraz port, w moim przypadku jest to port linux-owy, na platformie Windows będzie to jakiś COMx.

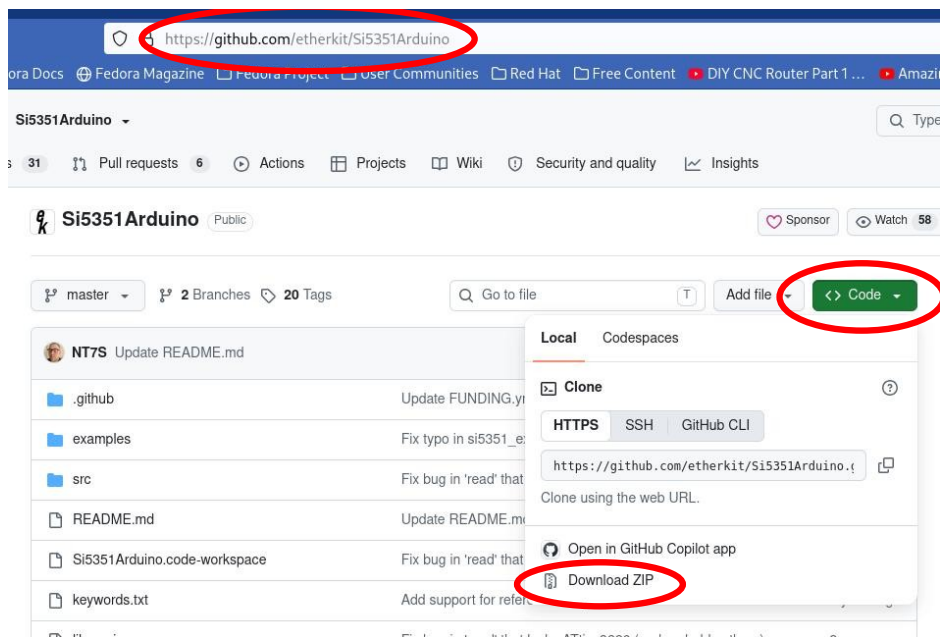
2. Instalacja biblioteki Etherkit

Instalację można wykonać na dwa sposoby:

- wybór i instalacja biblioteki za pomocą manager-a bibliotek Arduino



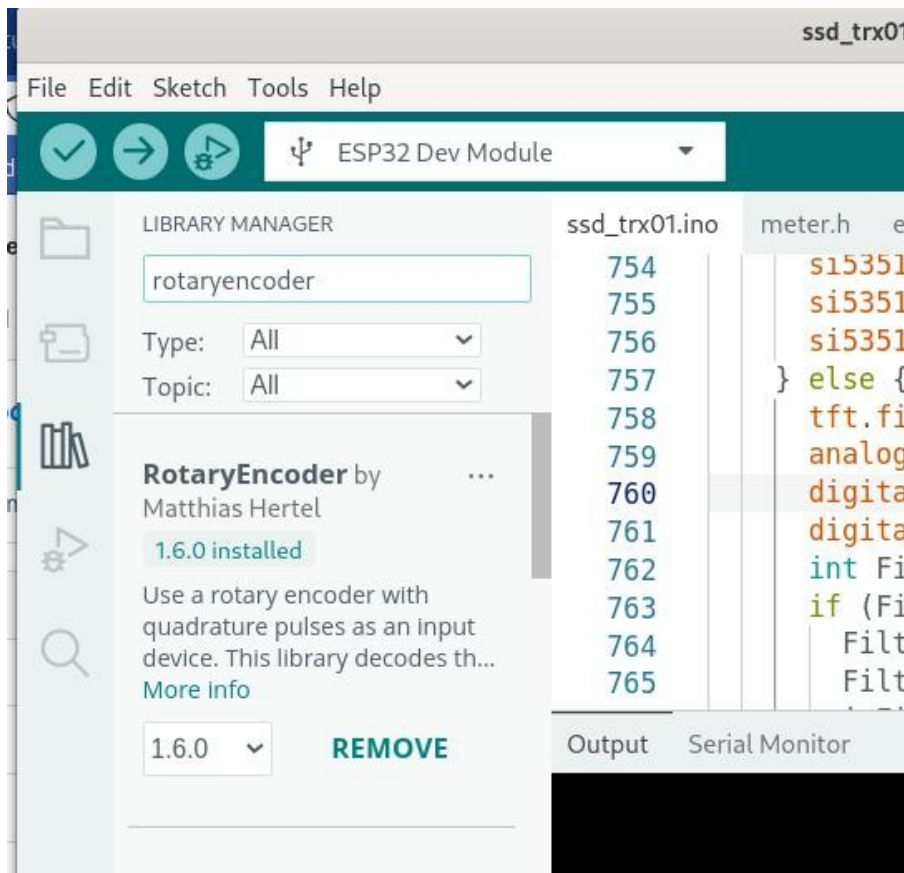
- wybór i instalacja biblioteki ze strony Github przez wybranie adresu strony a następnie przez klawisz CODE download pliku zip i jego instalację: **Sketch** → **Include Library** → **Add .ZIP file** → **wskazanie ściągniętego pliku .zip**



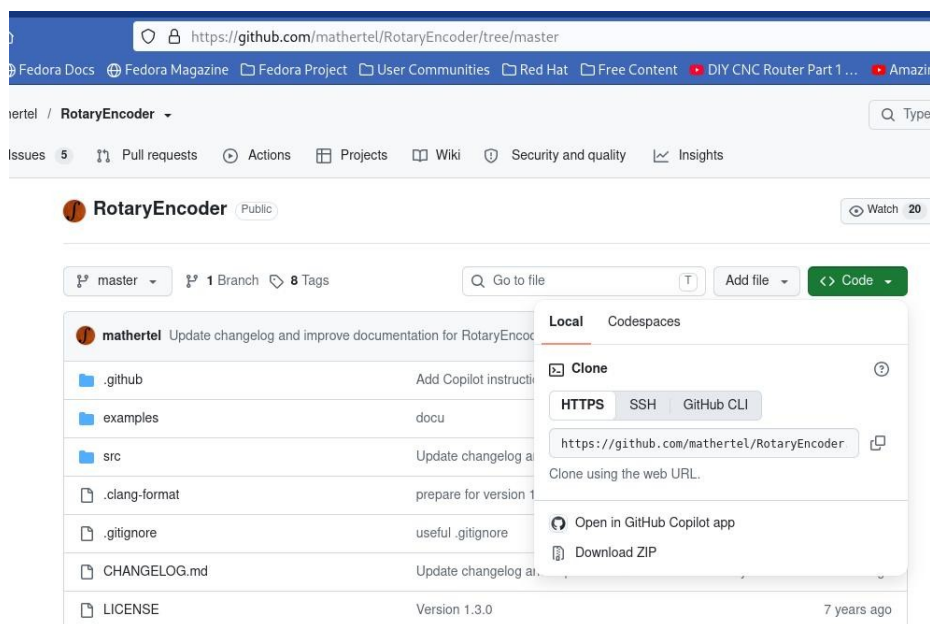
2. Instalacja biblioteki Rotaryencoder

Instalację można wykonać na dwa sposoby:

- wybór i instalacja biblioteki za pomocą manager-a bibliotek Arduino:

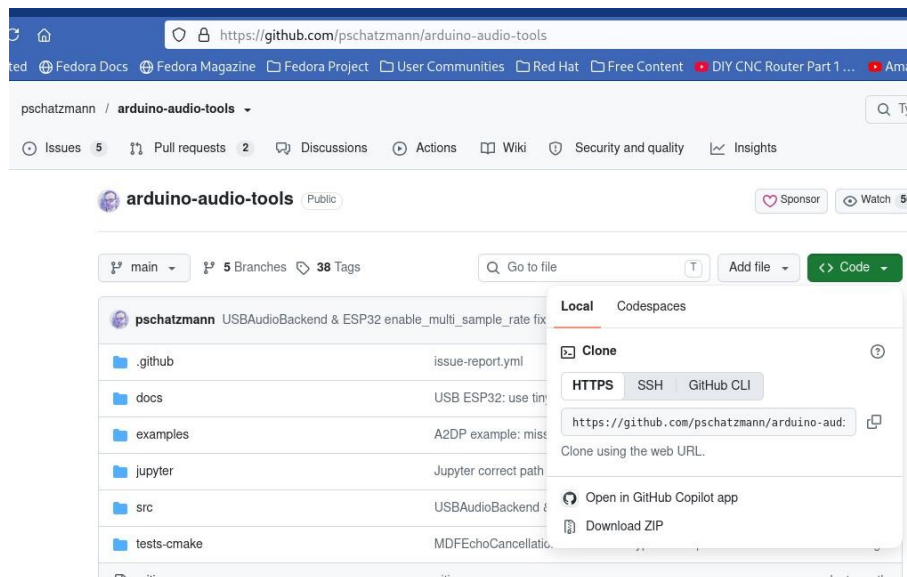


- wybór i instalacja biblioteki ze strony Github przez wybranie adresu strony a następnie przez klawisz CODE download pliku zip i jego instalację: **Sketch** → **Include Library** → **Add .ZIP file** → **wskazanie ściągniętego pliku .zip**



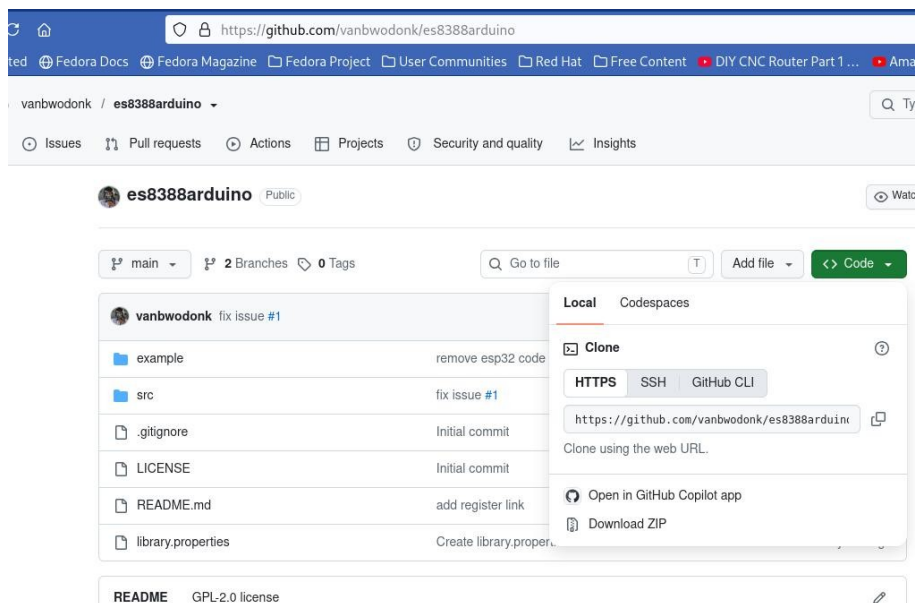
3. Instalacja bibliotek Arduino Audio Tools

- wybór i instalacja biblioteki ze strony Github przez wybranie adresu strony a następnie przez klawisz CODE download pliku zip i jego instalację: **Sketch** → **Include Library** → **Add .ZIP file** → **wskazanie ściągniętego pliku .zip**



4. Instalacja biblioteki es8388

- wybór i instalacja biblioteki ze strony Github przez wybranie adresu strony a następnie przez klawisz CODE download pliku zip i jego instalację: **Sketch** → **Include Library** → **Add .ZIP file** → **wskazanie ściągniętego pliku .zip**



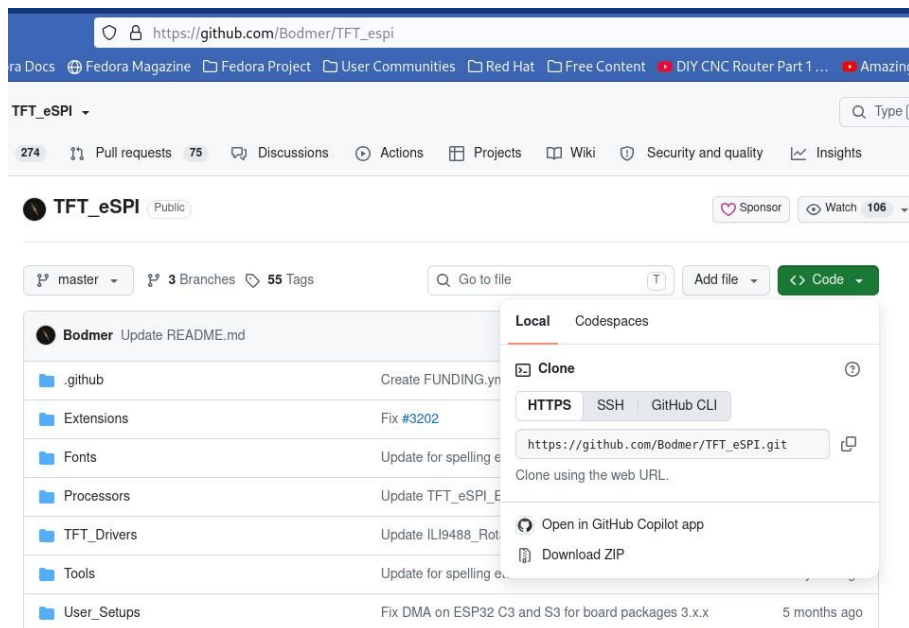
Niezbędne do pracy pliki można też ściągnąć z mojej strony i wgrać je w głównym katalogu projektu (sketch-a):

<https://lx-net.pl/img/es8388.cpp>

<https://lx-net.pl/img/es8388.h>

4. Instalacja biblioteki TFT ESPI

- wybór i instalacja biblioteki ze strony Github przez wybranie adresu strony a następnie przez klawisz CODE download pliku zip i jego instalację: **Sketch** → **Include Library** → **Add .ZIP file** → **wskazanie ściągniętego pliku .zip**



Do pracy z wyświetlaczem ili9341 należy zmodyfikować plik: **Katalog_Arduino** → **Libraries** → **TFTeSPI** → **User_Setup.h** (moja wersja tego pliku jest dostępna tu: https://lx-net.pl/img/User_Setup.h)

```
// User defined information reported by "Read_User_Setup" test & diagnostics example
#define USER_SETUP_INFO "User_Setup"

// Define to disable all warnings in library (can be put in User_Setup_Select.h)
// #define DISABLE_ALL_LIBRARY_WARNINGS

// =====
//
// Section 1. Call up the right driver file and any options for it
//
// =====

// Define SIM32 to invoke optimised processor support (only for SIM32)
// #define SIM32

// Defining the SIM32 board allows the library to optimise the performance
// for UNO compatible "MCUfriend" style shields
// #define NUCLEO_64_IET
// #define NUCLEO_144_IET

// SIM32 8-bit parallel only:
// If SIM32 Port A or B pins 0-7 are used for 8-bit parallel data bus bits 0-7
// then this will improve rendering performance by a factor of ~8x
// #define SIM_PORTA_DATA_BUS
// #define SIM_PORTB_DATA_BUS

// Tell the library to use parallel mode (otherwise SPI is assumed)
// #define IET_PARALLEL_8_BIT
// #define IET_PARALLEL_16_BIT // **** 16-bit parallel ONLY for RP2040 processor ****

// Display type - only define if RPi display
// #define RPI_DISPLAY_TYPE // 20MHz maximum SPI

// Only define one driver, the other ones must be commented out
#define ILI9341_DRIVER // Generic driver for common displays
// #define ILI9341_TFT_DRIVER // Alternative TFT9341 driver: see https://github.com/Bodmer/TFT_eSPI
```

Poniższe obrazy pokazują, które linie powinny być odkomentowane oraz jakie wartości dla pinów należy ustawić w kolejnych sekcjach pliku User_Setup.h


```
// #define TFT_SDA_READ // This option is for ESP32 ONLY, tested with ST7789 and GC9A01 display only

// For ST7735, ST7789 and ILI9341 ONLY, define the colour order IF the blue and red are swapped on your display
// Try ONE option at a time to find the correct colour order for your display

// #define TFT_RGB_ORDER TFT_RGB // Colour order Red-Green-Blue
#define TFT_RGB_ORDER TFT_BGR // Colour order Blue-Green-Red

// For M5Stack ESP32 module with integrated ILI9341 display ONLY, remove // in line below

// #define M5STACK

// For ST7789, ST7735, ILI9163 and GC9A01 ONLY, define the pixel width and height in portrait orientation
// #define TFT_WIDTH 80
// #define TFT_WIDTH 128
// #define TFT_WIDTH 172 // ST7789 172 x 320
// #define TFT_WIDTH 170 // ST7789 170 x 320
#define TFT_WIDTH 240 // ST7789 240 x 240 and 240 x 320
// #define TFT_HEIGHT 160
// #define TFT_HEIGHT 128
// #define TFT_HEIGHT 240 // ST7789 240 x 240
#define TFT_HEIGHT 320 // ST7789 240 x 320
// #define TFT_HEIGHT 240 // GC9A01 240 x 240

// For ST7735 ONLY, define the type of display, originally this was based on the
// colour of the tab on the screen protector film but this is not always true, so try
// out the different options below if the screen does not display graphics correctly.
```

```
// ##### EDIT THE PIN NUMBERS IN THE LINES FOLLOWING TO SUIT YOUR ESP32 SETUP #####
```

```
// For ESP32 Dev board (only tested with ILI9341 display)
// The hardware SPI can be mapped to any pins
```

```
#define TFT_MISO 18 //13 //4 //19
#define TFT_MOSI 19 //11 //5 //23
#define TFT_SCLK 23 //12 //6 //18
#define TFT_CS 21 //10 //7 // 15 // Chip select control pin
#define TFT_DC 22 //15 // 2 // Data Command control pin
// #define TFT_RST 19 //4 // Reset pin (could connect to RST pin)
```

```
#define TFT_RST -1 // Set TFT_RST to -1 if display RESET is connected to ESP32 board RST.
```

```
// For ESP32 Dev board (only tested with GC9A01 display)
// The hardware SPI can be mapped to any pins
```

```
/*#define TFT_MOSI 15 // In some display driver board, it might be written as "SDA" and so on.
#define TFT_SCLK 14
#define TFT_CS 5 // Chip select control pin
#define TFT_DC 27 // Data Command control pin
#define TFT_RST 33 // Reset pin (could connect to Arduino RESET pin)
#define TFT_BL 22 // LED back-light
```

```
#define TOUCH_CS 5 //16 //21 // Chip select pin (T_CS) of touch screen
```

```
/*#define TFT_WR 22 // Write strobe for modified Raspberry Pi TFT only
```

```
// For the M5Stack module use these #define lines
```

```
// The XPT2046 requires a lower SPI clock rate of 2.5MHz so we define that here:
```

```
#define SPI_TOUCH_FREQUENCY 2500000
```

```
// The ESP32 has 2 free SPI ports i.e. VSPI and HSPI, the VSPI is the default.
// If the VSPI port is in use and pins are not accessible (e.g. JIGQ T-Beam)
// then uncomment the following line:
```

```
#define USE_HSPI_PORT
```

```
/*#define USE_VSPI_PORT
```

```
// Comment out the following #define if "SPI Transactions" do not need to be
// supported. When commented out the code size will be smaller and sketches will
// run slightly faster, so leave it commented out unless you need it!
```

```
// Transaction support is needed to work with SD library but not needed with TFT_SdFat
// Transaction support is required if other SPI devices are connected.
```

```
// Transactions are automatically enabled by the library for an ESP32 (to use HAL mutex)
// so changing it here has no effect
```

```
#define SUPPORT_TRANSACTIONS
```