To świetne podejście! Analiza kodu „od kuchni” to najszybsza droga do zostania ekspertem. Twój projekt opiera się na **strukturze obiektowej**, co oznacza, że sterujemy ekranem jak „obiektem” (nazywamy go display).
Oto szczegółowy przewodnik po tym, co dzieje się w Twoim kodzie, podzielony na kluczowe sekcje:

1. Inicjalizacja i Biblioteki

Na samym początku masz deklaracje, które „mówią” Arduino, jak rozmawiać z ekranem:

- **#include <Wire.h>**: To biblioteka **I2C**. Ekran łączy się z Arduino za pomocą tylko dwóch przewodów (SDA i SCL). Ta biblioteka zajmuje się całą "komunikacją".
- **Adafruit_GFX oraz Adafruit_SSD1306**: To Twój "silnik graficzny".
    - GFX odpowiada za logikę rysowania (kropki, linie, tekst).
    - SSD1306 to sterownik konkretnie Twojego ekranu OLED.
- **Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);**: Tutaj tworzysz **obiekt** o nazwie display. Wszystkie dalsze komendy zaczynające się od display. to instrukcje wysyłane do tego konkretnego obiektu.

2. Mechanizm "Maszyny Stanów" (State Machine)

To najważniejszy element Twojej nowej logiki. Dzięki niemu urządzenie "pamięta", w jakim jest trybie:

- **int stanProgramu = 0;**: Zmienna globalna. Jeśli jest 0 – pokazujemy gwiazdy, jeśli 1 – powitanie.
- **digitalRead(BUTTON_PIN)**: Arduino sprawdza stan pinu cyfrowego.
    - Dzięki INPUT_PULLUP w setup(), pin jest domyślnie w stanie HIGH (napięcie 5V).
    - Kiedy naciskasz przycisk, zwierasz pin do GND, więc stan zmienia się na LOW.
    - Kod stanProgramu = !stanProgramu; to tzw. **negacja logiczna** – jeśli było 0, staje się 1 (i na odwrót).

3. Anatomia rysowania (Sekcja loop)

Wszystko, co widzisz na ekranie, dzieje się w tej kolejności w każdej milisekundzie:

1. **display.clearDisplay();**: Czyści wirtualną pamięć ekranu (czarny prostokąt). Jeśli tego nie zrobisz, nowe elementy będą rysowane na starych.
2. **display.drawPixel(x, y, WHITE);**: Rysuje pojedynczy punkt. To podstawa Twojej animacji gwiazd.
3. **display.setTextSize(1);**: Ustawia skalę czcionki. 1 to rozmiar standardowy, 2 to dwukrotne powiększenie.
4. **display.setCursor(x, y);**: Ustawia "pióro" w konkretnym miejscu. Pamiętaj: (0,0) to lewy górny róg.
5. **display.println("...");**: Wrzuca tekst do wirtualnej pamięci ekranu.
6. **display.display();**: **Kluczowa komenda!** Dopiero teraz Arduino wysyła zawartość pamięci przez przewody I2C do fizycznego wyświetlacza. Bez tego ekran zostanie pusty.

Kluczowe różnice w strukturze:

Komenda	Co robi?
begin()	"Budzi" ekran i sprawdza, czy żyje.
clearDisplay()	Czyści wirtualny bufor pamięci.
drawPixel()	Rysuje punkt w buforze.
println()	Dodaje tekst do bufora.
display()	Wysyła bufor na ekran (pokazuje efekt).

Ćwiczenie dla Ciebie (do samodzielnej analizy):

Jeśli chcesz poczuć jak to działa, spróbuj zmienić w kodzie display.drawRect(0, 0, 128, 64, WHITE); na display.drawRect(10, 10, 108, 44, WHITE);.
Co się stanie? Ramka się skurczy. Dzięki temu zrozumiesz, że te cztery cyfry to odpowiednio: *X-start, Y-start, Szerokość, Wysokość*.
Czy taka analiza "element po elemencie" jest dla Ciebie jasna? Chcesz, żebyśmy jeszcze mocniej zgłębili któryś z tych punktów (np. jak dokładnie działa ten matematyczny "ruch" gwiazd)?