

## PROGRAM SZKOLENIA – „Podstawy Programowania Mikrokontrolerów dla dorosłych”

**Miejsce realizacji:** Fablab Chrzanów, ul. Janiny Woynarowskiej 1, 32-500 Chrzanów

**Liczebność grupy:** maksymalnie 8 osób

**Forma wsparcia:** warsztaty praktyczne + krótkie wprowadzenia teoretyczne

**Czas trwania:** 4 spotkania po 4 godziny (łącznie 16 godziny), raz w tygodniu

### Cel główny:

- Obsługiwać przyrządy pomiarowe: multimetr, oscyloskop, generator funkcyjny
- Programować mikrokontrolery ESP32, Arduino w środowiskach Arduino IDE
- Konfigurować protokoły komunikacyjne: UART, I2C, SPI
- Podłączać i obsługiwać popularne moduły i czujniki

## ETAP I ZAKRES

### Grupa docelowa

Dorośli (18+) zainteresowani elektroniką, programowaniem i systemami wbudowanymi. Kurs skierowany do programistów, elektryków, automatyków i hobbystów chcących nabyć praktyczne kompetencje w zakresie programowania mikrokontrolerów.

### Zakres tematyczny podlegający ocenie (kompetencja)

'Podstawy programowania mikrokontrolerów – obejmujące:

- zasady bezpiecznej pracy z elektroniką niskonapięciową i BHP,
- znajomość architektury mikrokontrolerów Arduino, ESP32 i STM32,
- programowanie GPIO, PWM, ADC i interfejsów komunikacyjnych (UART, I2C, SPI),
- obsługę popularnych czujników: DHT22, HC-SR04, BMP280, INA219,

## ETAP II WZORZEC (EFEKTY UCZENIA SIĘ)

### WIEDZA — uczestnik/uczestniczka wie:

- co to jest mikrokontroler, czym różni się od komputera ogólnego przeznaczenia i gdzie jest stosowany
- jaką architekturę mają platformy: Arduino Uno (AVR 8-bit, 16MHz), ESP32 (Xtensa LX6 32-bit, 240MHz), STM32 (ARM Cortex-M4, 180MHz)
- co to są piny GPIO i jak skonfigurować je jako wejścia cyfrowe (z pull-up/pull-down) i wyjścia cyfrowe

- co to jest PWM (Pulse Width Modulation) i do jakich zastosowań go używamy: jasność LED, pozycja serwa, prędkość silnika
- jak działają interfejsy: UART/Serial (dwa przewody, asynchroniczny), I2C (SDA/SCL, adresy 7-bit), SPI (4 przewody, synchroniczny)
- co to jest ADC (przetwornik A/C) – rozdzielczość 10-bit (Arduino) i 12-bit (ESP32, STM32), napięcie referencyjne, błędy kwantyzacji
- jak działa timer sprzętowy, do czego służy i jak generować precyzyjne przerwania czasowe (1 ms, 100 ms)
- jak działa WiFi w ESP32: tryb Station, AP, WPA2, procedura łączenia, obsługa rozłączenia
- jak działają czujniki: DHT22 (1-wire, temp  $\pm 0,5^{\circ}\text{C}$ , wilg  $\pm 2\%$ ), HC-SR04 (ultradźwięki, 2–400 cm), BMP280 (I2C, ciśnienie/temp)

### UMIEJĘTNOŚCI — uczestnik/uczestniczka potrafi:

- napisać szkic Arduino (setup + loop) i wgrać go na płytkę przez Arduino IDE lub PlatformIO przez USB
- sterować diodą LED przez GPIO i serwomechanizmem przez PWM – regulacja jasności i kąta obrotu
- odczytywać stan przycisku z obsługą debouncingu programowego (millis()) i sprzętowego (kondensator)
- podłączyć i obsługiwać czujniki DHT22 (biblioteka Adafruit), HC-SR04, BMP280 (I2C) – odczyt i parsowanie danych
- skonfigurować Serial Monitor (9600/115200 baud) do debugowania i przesyłania danych do komputera
- przeskanować magistralę I2C i odczytywać rejestry czujnika pod znany adres
- skonfigurować WiFi ESP32 w trybie Station: SSID, hasło, oczekiwanie na IP, obsługa błędów łączenia
- skonfigurować ADC 12-bit na STM32 z DMA circular i timerem TIM2 wyzwalającym w STM32CubeIDE

### KOMPETENCJE SPOŁECZNE — uczestnik/uczestniczka:

- przestrzega zasad bezpiecznej pracy z elektroniką: napięcia maks. 5V/3,3V, polaryzacja, ESD, prądy max GPIO
- stosuje systematyczne podejście do debugowania: izolacja problemu, hipoteza, test, weryfikacja przez Serial Monitor
- pisze czytelny kod: nazewnictwo zmiennych (camelCase, UPPER dla stałych), komentarze funkcji, modularność
- samodzielnie korzysta z dokumentacji (datasheet, API reference), forów i repozytoriów GitHub
- dba o porządek na stanowisku, zabezpiecza połączenia na płytce stykowej, prawidłowo przechowuje moduły

- dokonuje samooceny jakości firmware: niezawodność, wydajność, czytelność i wyciąga wnioski

## TREŚCI PROGRAMOWE

Nr	Tytuł spotkania	Treści programowe
S1	<b>Wprowadzenie do mikrokontrolerów</b>	<ul style="list-style-type: none"> <li>• Test początkowy</li> <li>• Zastosowania mikrokontrolerów: AGD, automotive, przemysł, IoT – gdzie jest mikrokontroler?</li> <li>• Porównanie platform: Arduino Uno (AVR), ESP32 (Xtensa), STM32 (ARM Cortex-M) – różnice architektoniczne</li> <li>• Środowisko Arduino IDE: instalacja, wybór płytki, port COM, Blink jako pierwszy program</li> <li>• GPIO digital: pinMode, digitalWrite, digitalRead – sterowanie LED i odczyt przycisku</li> <li>• BHP: napięcia bezpieczne, polaryzacja zasilania, ESD, prądy maksymalne pinów GPIO</li> </ul>
S2	<b>Wyjścia analogowe i interfejsy komunikacyjne</b>	<ul style="list-style-type: none"> <li>• PWM: analogWrite, wypełnienie 0–255, zastosowania – jasność LED, serwomechanizm</li> <li>• ADC: analogRead 10-bit (Arduino) – skalowanie do wartości fizycznych (np. temp NTC)</li> <li>• UART/Serial: Serial.begin, Serial.print, Serial.println – debugging i telemetria</li> <li>• I2C: Wire.begin, Wire.beginTransmission, Wire.write/read – skaner adresów, odczyt rejestru</li> <li>• SPI: podstawy, konfiguracja jako Master, zastosowania (wyświetlacze, karty SD)</li> <li>• Ćwiczenie: czujnik DHT22 (temperatura + wilgotność) przez 1-wire – odczyt w Serial Monitor</li> </ul>
S3	<b>Czujniki i systemy wbudowane</b>	<ul style="list-style-type: none"> <li>• HC-SR04: zasada działania (echo ultradźwiękowe), schemat podłączenia, kod pomiaru odległości</li> <li>• BMP280 przez I2C: biblioteka Adafruit, odczyt ciśnienia i temperatury, obliczanie wysokości</li> <li>• Przerwania: attachInterrupt, ISR – ograniczenia, volatile, przykład RPM miernika</li> <li>• Timer sprzętowy: millis() vs delay() – dlaczego millis() jest preferowane w embedded</li> <li>• Debouncing: przyczyny drgań styku, debouncing programowy (20 ms okno) i sprzętowy (RC)</li> <li>• Ćwiczenie: stacja pogodowa – DHT22 + BMP280 + HC-SR04 → wyświetlanie w Serial Monitor</li> </ul>
S4	<b>ESP32 – WiFi i MQTT</b>	<ul style="list-style-type: none"> <li>• Architektura ESP32: dual-core, WiFi, BT, 18 kanałów ADC 12-bit, GPIO touch</li> <li>• WiFi Station: WiFi.begin, WiFi.status, timeout, reconnect loop</li> <li>• ADC 12-bit + DMA circular: konfiguracja, Transfer Complete interrupt, bufor kołowy</li> <li>• Test końcowy (post-test) – wiedza teoretyczna</li> <li>• Ocena praktyczna – demonstracja działającego systemu IoT</li> <li>• Samoocena uczestnika + podsumowanie nabytych kompetencji</li> </ul>

## ETAP III KRYTERIA I METODY WERYFIKACJI EFEKTÓW UCZENIA SIĘ

### Metody weryfikacji teoretycznej

- Test końcowy (post-test) – architektura MCU, GPIO, PWM, ADC, I2C, BHP
- Pytania ustne podczas zajęć praktycznych
- Ocena jakości napisanego firmware: struktura, komentarze, obsługa błędów

### Metody weryfikacji praktycznej

- Ocena działania programu Arduino: LED, przycisk z debouncem, odczyt czujnika
- Ocena firmware ESP32: połączenie WiFi
- Ocena konfiguracji STM32: ADC z DMA, wyniki w Serial Monitor z poprawnymi wartościami
- Samoocena uczestnika

### Uczestnik nabywa kompetencje, jeśli:

- Uzyska min. 80% poprawnych odpowiedzi w post-teście
- Napisze i uruchomi firmware ESP32 z WiFi
- Skonfiguruje Mikrokontroler ADC z DMA i wyświetli wyniki w Serial Monitor
- Przestrzega zasad BHP przez cały czas trwania kursu
- Uczestniczył w min. 80% godzin zajęciowych (min. 20 z 24 h)