PROGRAM SZKOLENIA – „Podstawy Programowania Mikrokontrolerów"

Miejsce realizacji: Fablab Chrzanów, ul. Janiny Woynarowskiej 1, 32-500 Chrzanów
Liczebność grupy: maksymalnie 6 osób
Forma wsparcia: warsztaty praktyczne + krótkie wprowadzenia teoretyczne
Czas trwania: 6 spotkań po 4 godziny (łącznie 24 godziny), raz w tygodniu

Cel główny:

- Obsługiwać przyrządy pomiarowe: multimetr, oscyloskop, generator funkcyjny
- Programować mikrokontrolery ESP32, Arduino i STM32 w środowiskach Arduino IDE i STM32CubeIDE
- Konfigurować protokoły komunikacyjne: UART, I2C, SPI
- Podłączać i obsługiwać popularne moduły i czujniki
- Realizować projekt IoT z ESP32 i komunikacją MQTT/WiFi

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 i tworzenia urządzeń IoT. Zalecana podstawowa znajomość programowania lub elektroniki.

Zakres tematyczny podlegający ocenie (kompetencja)

'Podstawy programowania mikrokontrolerów i systemów IoT – 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,
- konfigurację WiFi i protokołu MQTT na ESP32 – wysyłanie danych do chmury,
- konfigurację STM32 z HAL: ADC/DMA, timery sprzętowe, watchdog,
- implementację Modbus TCP Slave na ESP32 – integracja z systemami PLC.

Projekt „Utworzenie innowacyjnych przestrzeni typu Fablab na terenie Małopolski Zachodniej wraz z organizacją działań mobilnych" dofinansowany ramach priorytetu 8 Fundusze europejskie dla sprawiedliwej transformacji Małopolski Zachodniej 8.2 Edukacja dla transformacji, typ A: Tworzenie przestrzeni typu fablab programu Fundusze Europejskie dla Małopolski 2021-2027. Nr projektu: FEMP.08.02-IP.01-0017/23

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)
- co to jest przerwanie (interrupt) – ISR, priorytety, różnica wobec pollingu, volatile zmienne
- jak działa WiFi w ESP32: tryb Station, AP, WPA2, procedura łączenia, obsługa rozłączenia
- co to jest protokół MQTT: broker, temat (topic), QoS 0/1/2, retain, Last Will, publikacja i subskrypcja
- jak działają czujniki: DHT22 (1-wire, temp ±0,5°C, wilg ±2%), HC-SR04 (ultradźwięki, 2–400 cm), BMP280 (I2C, ciśnienie/temp)
- co to jest watchdog timer – działanie, konfiguracja, znaczenie w systemach embedded bez nadzoru człowieka

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 (Wire.h scanner) i odczytywać rejestry czujnika pod znany adres
- skonfigurować WiFi ESP32 w trybie Station: SSID, hasło, oczekiwanie na IP, obsługa błędów łączenia

- zaimplementować klienta MQTT na ESP32 (biblioteka PubSubClient): connect, publish JSON, subscribe, callback
- skonfigurować ADC 12-bit na STM32 z DMA circular i timerem TIM2 wyzwalającym w STM32CubeIDE
- zaimplementować Modbus TCP Slave na ESP32 (biblioteka ModbusIP): mapa rejestrów, skalowanie wartości
- napisać firmware z watchdogiem (ESP.wdtFeed / IWDG) i obsługą awarii WiFi z automatycznym wznowieniem
- zmierzyć latencję od zdarzenia do danych w dashboardzie i zoptymalizować kod pod kątem czasu odpowiedzi

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
- współpracuje przy integracji firmware z systemami nadrzędnymi (PLC Modbus, MQTT dashboard)
- dokonuje samooceny jakości firmware: niezawodność, wydajność, czytelność i wyciąga wnioski
- rozumie konsekwencje błędów w embedded: utrata danych, crash urządzenia, zagrożenie bezpieczeństwa

TREŚCI TREŚCI PROGRAMOWE

Nr	Tytuł spotkania	Treści programowe
S1	**Wprowadzenie do mikrokontrolerów**	• Test początkowy • Zastosowania mikrokontrolerów: AGD, automotive, przemysł, IoT – gdzie jest mikrokontroler? • Porównanie platform: Arduino Uno (AVR), ESP32 (Xtensa), STM32 (ARM Cortex-M) – różnice architektoniczne • Środowisko Arduino IDE: instalacja, wybór płytki, port COM, Blink jako pierwszy program • GPIO digital: pinMode, digitalWrite, digitalRead – sterowanie LED i odczyt przycisku • BHP: napięcia bezpieczne, polaryzacja zasilania, ESD, prądy maksymalne pinów GPIO
S2	**Wyjścia analogowe i interfejsy komunikacyjne**	• PWM: analogWrite, wypełnienie 0–255, zastosowania – jasność LED, serwomechanizm • ADC: analogRead 10-bit (Arduino) – skalowanie do wartości fizycznych (np. temp NTC) • UART/Serial: Serial.begin, Serial.print, Serial.println – debugging i

		telemetria • I2C: Wire.begin, Wire.beginTransmission, Wire.write/read – skaner adresów, odczyt rejestru • SPI: podstawy, konfiguracja jako Master, zastosowania (wyświetlacze, karty SD) • Ćwiczenie: czujnik DHT22 (temperatura + wilgotność) przez 1-wire – odczyt w Serial Monitor
S3	**Czujniki i systemy wbudowane**	• HC-SR04: zasada działania (echo ultradźwiękowe), schemat podłączenia, kod pomiaru odległości • BMP280 przez I2C: biblioteka Adafruit, odczyt ciśnienia i temperatury, obliczanie wysokości • Przerwania: attachInterrupt, ISR – ograniczenia, volatile, przykład RPM miernika • Timer sprzętowy: millis() vs delay() – dlaczego millis() jest preferowane w embedded • Debouncing: przyczyny drgań styku, debouncing programowy (20 ms okno) i sprzętowy (RC) • Ćwiczenie: stacja pogodowa – DHT22 + BMP280 + HC-SR04 → wyświetlanie w Serial Monitor
S4	**ESP32 – WiFi i MQTT**	• Architektura ESP32: dual-core, WiFi, BT, 18 kanałów ADC 12-bit, GPIO touch • WiFi Station: WiFi.begin, WiFi.status, timeout, reconnect loop • Protokół MQTT: broker Mosquitto, topiki hierarchiczne, QoS 0/1/2, retain, Last Will • Biblioteka PubSubClient: connect, publish(topic, JSON), subscribe, callback • Format JSON: struktura, serializacja przez ArduinoJson • Ćwiczenie: ESP32 publikuje temp/wilg/odległość na broker MQTT co 10 s – weryfikacja w MQTT Explorer
S5	**STM32 i Modbus TCP Slave**	• STM32CubeIDE: tworzenie projektu, konfiguracja GPIO, USART, ADC w CubeMX • ADC 12-bit + DMA circular: konfiguracja, Transfer Complete interrupt, bufor kołowy • Timer TIM2: prescaler, period, wyzwalanie ADC – obliczanie dla 100 Hz próbkowania • Watchdog IWDG: konfiguracja, IWDG_Refresh w pętli, test – co się dzieje bez odświeżania • Modbus TCP Slave na ESP32 (biblioteka ModbusIP): rejestry 40001–40010, skalowanie • Ćwiczenie: ESP32 Modbus Slave – PLC odczytuje temperaturę z rejestru 40001 w TIA Portal Watch Table
S6	**Projekt uczestnika + walidacja**	• Napisanie kompletnego firmware ESP32: WiFi + MQTT + Modbus Slave + watchdog • Integracja z dashboardem Node-RED lub MQTT Explorer – weryfikacja danych live • Test latencji: sensor → ESP32 → MQTT broker → dashboard (cel: < 1 s) • Test końcowy (post-test) – wiedza teoretyczna • Ocena praktyczna – demonstracja działającego systemu IoT • Samoocena uczestnika + podsumowanie nabytych kompetencji

ETAP III KRYTERIA I METODY WERYFIKACJI EFEKTÓW UCZENIA SIĘ

Metody weryfikacji teoretycznej

- Test końcowy (post-test) – architektura MCU, GPIO, PWM, ADC, I2C, MQTT, Modbus, 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, publikacja MQTT, dane widoczne w MQTT Explorer
- Ocena konfiguracji STM32: ADC z DMA, wyniki w Serial Monitor z poprawnymi wartościami
- Ocena Modbus TCP Slave: rejestr 40001 odczytany przez PLC lub MQTT Explorer
- Pomiar latencji end-to-end: wynik ≤ 1000 ms
- Samoocena uczestnika

Uczestnik nabywa kompetencje, jeśli:

- Uzyska min. 80% poprawnych odpowiedzi w post-teście
- Napisze i uruchomi firmware ESP32 z WiFi + MQTT + Modbus Slave działającymi jednocześnie
- Skonfiguruje STM32 ADC z DMA i wyświetli wyniki w Serial Monitor
- Zmierzy latencję end-to-end i osiągnie wynik ≤ 1000 ms
- Przestrzega zasad BHP przez cały czas trwania kursu
- Uczestniczył w min. 80% godzin zajęciowych (min. 20 z 24 h)