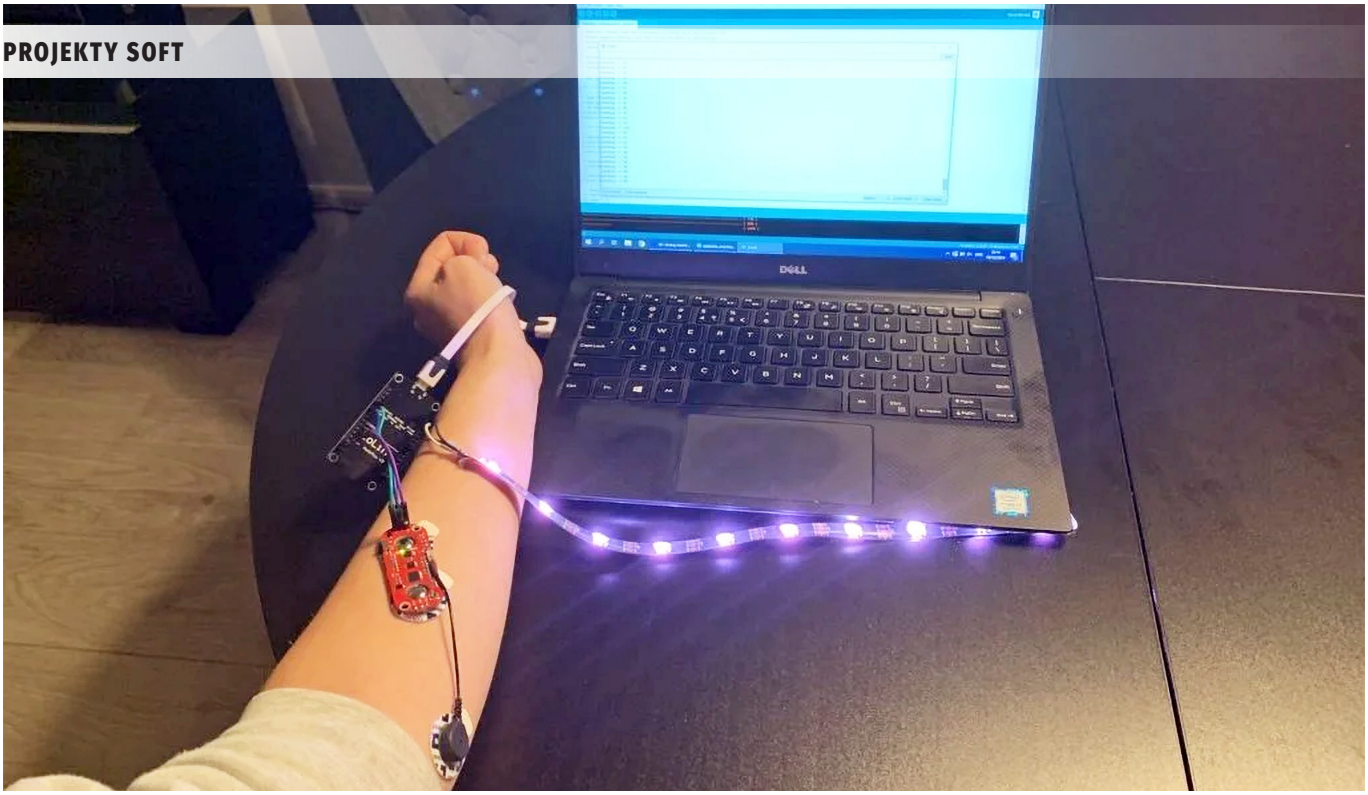


Sensor pracy mięśni z wizualizacją w sieci Web i na diodach LED RGB

Nazwa tego projektu początkowo może wydawać się nieco dziwna, ale poniższy system właśnie tak działa – steruje diodami LED RGB za pomocą impulsów elektrycznych, które docierają do naszych mięśni przez układ nerwowy. Działanie takiego urządzenia jest możliwe dzięki temu, że nasz „system sterowania”, podobnie jak mikrokontrolery, używa elektryczności.

Opisany poniżej układ bazuje na module do elektromiografii **MyoWare**. Sensor ten został zaprojektowany do pomiaru niewielkich sygnałów elektrycznych, jakie są obecne w naszych mięśniach, gdy to się poruszają. Na wyjściu sensora powstaje sygnał analogowy, który można mierzyć za pomocą dowolnego mikrokontrolera.

Elektromiografia (EMG) mierzy odpowiedź mięśni lub aktywność elektryczną w odpowiedzi na stymulację mięśnia przez układ nerwowy. Badanie to służy zazwyczaj do wykrywania nieprawidłowości nerwowo-mięśniowych. Podczas testu jedną lub więcej małych igieł (zwanych również elektrodami) wprowadza się przez skórę do mięśnia. W przypadku opisanego modułu nie ma potrzeby wprowadzać elektrod do mięśnia, wystarczy je nakleić na skórę.

Potrzebne elementy

Do budowy opisywanego urządzenia będą potrzebne następujące elementy:

- Moduł sensora mięśniowego MyoWare,
- Elektrody do EMG,
- Moduł NodeMCU (z mikrokontrolerem ESP8266),
- Pasek LED-owy Neopixel,
- Kable do podłączenia paska LED-owego i kabel microUSB.

Dodatkowo, na komputerze, na którym będziemy kompilować kod programu w Arduino IDE do tego urządzenia, oprócz samego środowiska programistycznego dla Arduino musimy zainstalować następujące biblioteki:

- Adafruit io Arduino (w wersji co najmniej 2.3.0),
- Adafruit Neopixel (wersja 1.2.5 i wyżej),
- Adafruit MQTT (1.0.3),
- Arduino HttpClients (0.4.0).

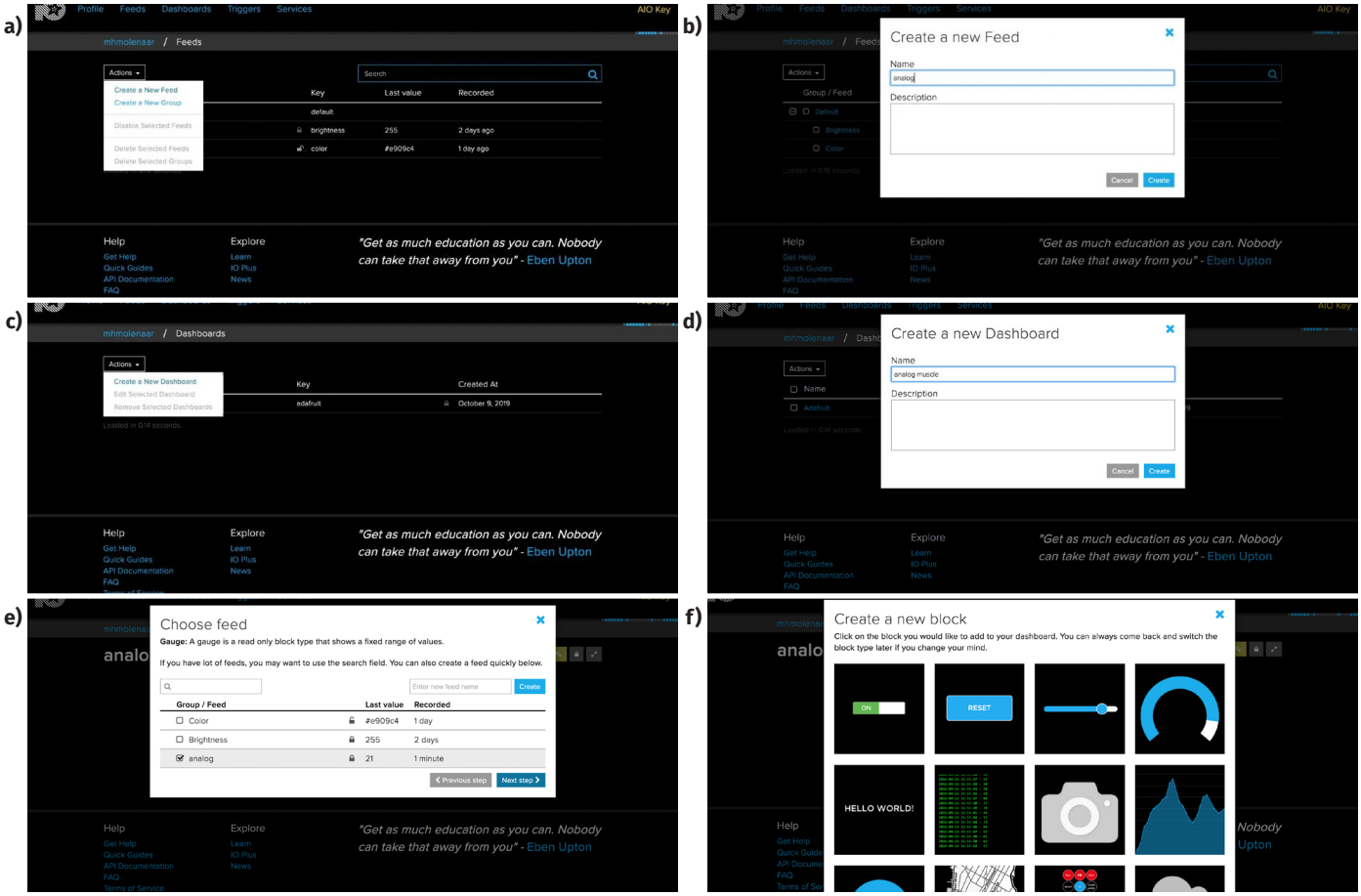
Jeśli nie posiadasz w swoim IDE zainstalowanych takich bibliotek, można je z łatwością doinstalować z poziomu Arduino IDE, klikając po kolei *Szkic* → *Dołącz bibliotekę* → *Zarządzaj bibliotekami*. Tam wystarczy wpisać nazwę biblioteki, aby móc ją odnaleźć i zainstalować

Krok 1: Podłączenie sensora MyoWare do NodeMCU

Jeśli nie jesteś zaznajomiony z sensorem MyoWare, realizację tego projektu warto rozpocząć od zrozumienia, w jaki sposób sensor jest podłączony do Arduino. Sposób przygotowania opisany został na stronie Adafruit ***http://bit.ly/31PcwpP***.

Do podłączenia sensora potrzebne są zaledwie trzy przewody – dwa do zasilania (VCC i masa) oraz jeden dla sygnału wyjściowego. Powinny to być dłuższe kable zakończone złączami do goldpinów modułu NodeMCU. Dodatkowo musimy przylutować krótki kabelek zakończony elektrodą.

W celu przetestowania modułu podłączamy wyjście sygnału EMG do wejścia analogowego A0 mikrokontrolera. Z menu w Arduino IDE



Rysunek 1. Konfiguracja Adafruit IO

należy wybrać przykład *01.Basics* → *AnalogReadSerial*. Szkic ten odczytuje wartość analogową z pinu A0 i przesyła ją poprzez port szeregowy. Jeśli po załadowaniu szkicu zobaczymy na ekranie monitora portu szeregowego wartości zmieniające się wraz z ruchem mięśni, do których podłączona jest elektroda MyoWare, to znaczy, że wszystko poprawnie podłączyliśmy.

Krok 2: Konfiguracja Adafruit IO

Pierwszym krokiem jest zalogowanie się lub utworzenie konta na portalu Adafruit IO. Wchodzimy na stronę www.io.adafruit.com i tam logujemy się lub tworzymy nowe konto użytkownika, którego będziemy używali w dalszej części projektu. Po zalogowaniu się do naszego konta możemy utworzyć nowe źródło danych (kanał) – *Feed* – tak jak pokazano na **rysunku 1a** oraz **1b**. Feed nazywamy *Analog*, ponieważ będzie obsługiwał wejście analogowe. Jeśli chcemy, możemy dodać do niego także opis, objaśniający, co to za wejście.

Jeśli chcemy prezentować dane z wejścia analogowego w Adafruit IO, musimy dodać i skonfigurować odpowiedni panel, gdzie będziemy je pokazywali. W tym celu, jak pokazano na **rysunku 1c** i **1d**, tworzymy nowy *Dashboard*, który nazywamy dowolnie. Na tym panelu możemy dodać dowolne wizualizacje tych danych, pamiętając tylko, by jako źródło danych ustawić skonfigurowany wcześniej feed o nazwie *Analog* (patrz **rysunek 1e**). Wybierzmy wskaźnik o nazwie *Gauge* i skonfigurujmy zakres danych od 0 (minimum) do 1024 (maksimum) – takie wartości zwraca nasz sensor. W katalogu *Adafruit IO* znajdziemy wiele różnych metod wizualizacji danych, część z nich została pokazana na **rysunku 1f.**

Krok 3: Wejście analogowe w Arduino

W kroku pierwszym podłączyliśmy i przetestowaliśmy sensor pracy mięśni MyoWare, teraz przejdziemy do oprogramowania go na mikrokontrolerze, z użyciem biblioteki i dashboarda, który wygenerowaliśmy w kroku 2. Nasz program będzie łączył się z siecią i z API Adafruit IO, gdzie będzie wysyłał zbierane dane. Pozwoli to na wizualizowanie ruchu mięśni i prezentację efektów w sieci. Po upewnieniu się, że mamy zainstalowane wszystkie wymienione powyżej biblioteki, możemy uruchomić przykładowy szkic, na którym zbudujemy nasz program. W tym celu z menu z przykładami wybieramy: *Adafruit IO Arduino* → *adafruitio_08_analog_in*, jak pokazano na **rysunku 2a**. Musimy teraz skonfigurować w tym przykładowym szkicu szereg rzeczy, aby działał z naszym kontem Adafruit IO.

Aby skonfigurować ustawienia sieciowe, kliknij kartę *config.h* na szkicu. Musimy wpisać tam swoją nazwę użytkownika Adafruit IO w definicji IO_USERNAME, a klucz do IO Adafruit w definicji IO_KEY. Nazwę użytkownika i klucz znaleźć można w prawym górnym rogu ekranu na stronie Arduino IO (**rysunek 2b**):

```
#define IO_USERNAME    „your_username”
#define IO_KEY         „your_key”
```

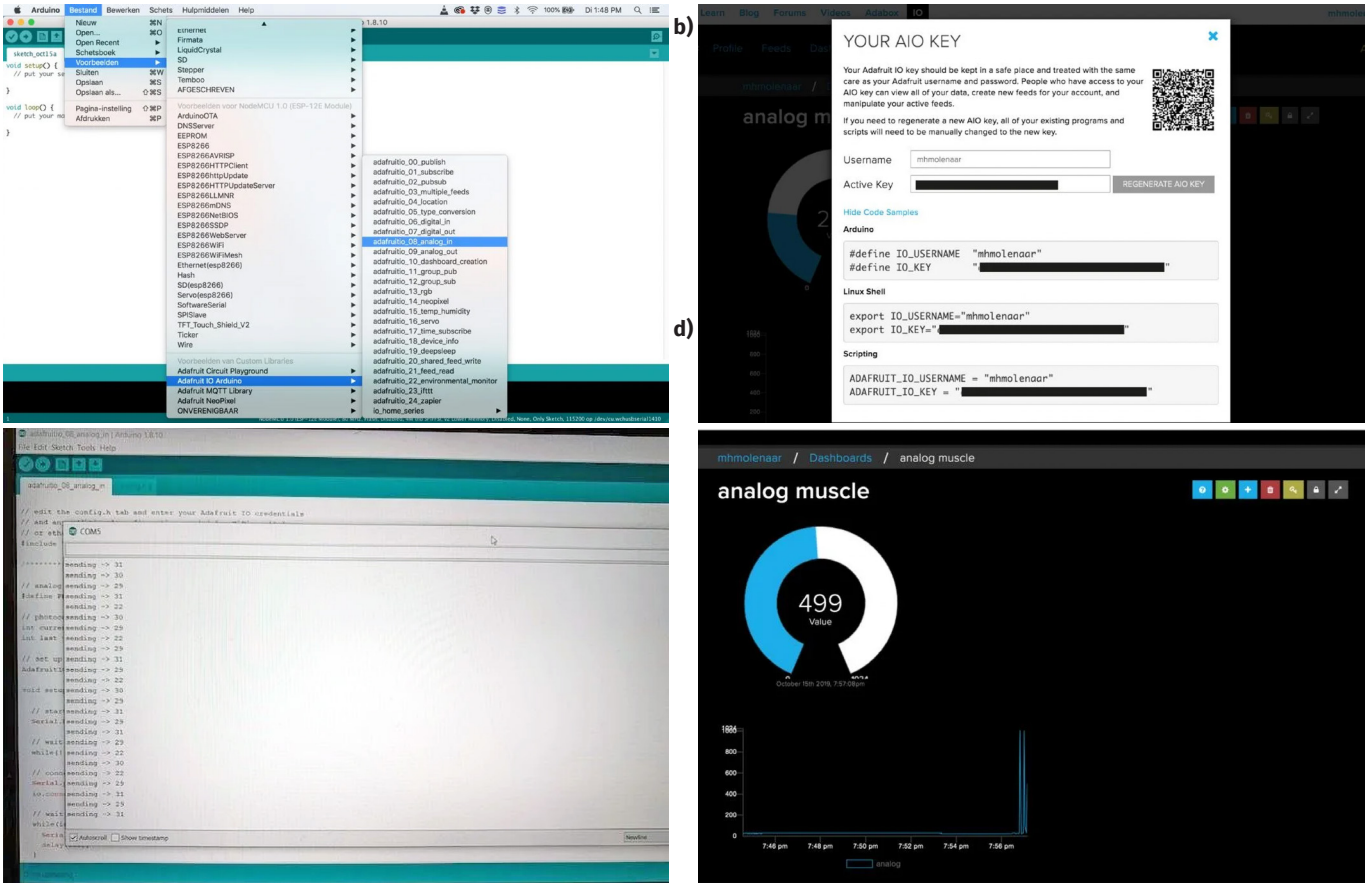
Następnie możemy skonfigurować połączenie NodeMCU z naszą siecią Wi-Fi. W tym celu musimy uzupełnić w pliku *config.h* parametry WIFI_SSID oraz WIFI_PASS, opisujące, odpowiednio, nazwę naszej sieci domowej oraz hasło, jakiego NodeMCU ma używać podczas podłączania się do Wi-Fi.

```
#define WIFI_SSID      „your_ssid”
#define WIFI_PASS      „your_pass”
```

Po wprowadzeniu powyższych zmian możemy przetestować system. Jeśli podłączyliśmy sygnał z sensora do wejścia analogowego A0, to nie musimy nic zmieniać. Możemy zmienić nazwę zmiennej, przechowującej numer pinu, by zwiększyć czytelność kodu. Zmieniamy PHOTOCELL_PIN na MUSCLESENSOR_PIN. Robimy tak w dwóch miejscach, gdzie występuje odwołanie do tej wartości:

```
// analog pin 0
#define PHOTOCELL_PIN A0
```

W ostatniej linijce kodu znajduje się opóźnienie – jest ono ustawione domyślnie na 1000 milisekund, ale zmieniamy na 2000 milisekund,



Rysunek 2. Szkic obsługujący Adafruit IO oraz dane potrzebne do zalogowania do systemu

z uwagi na ograniczenie przepustowości Adafruit IO w wersji podstawowej, z jakiej korzystamy (szybsze przetwarzanie danych wymaga stosownej licencji).

```
// wait one second (1000 milliseconds == 1 second)
delay(2000);
```

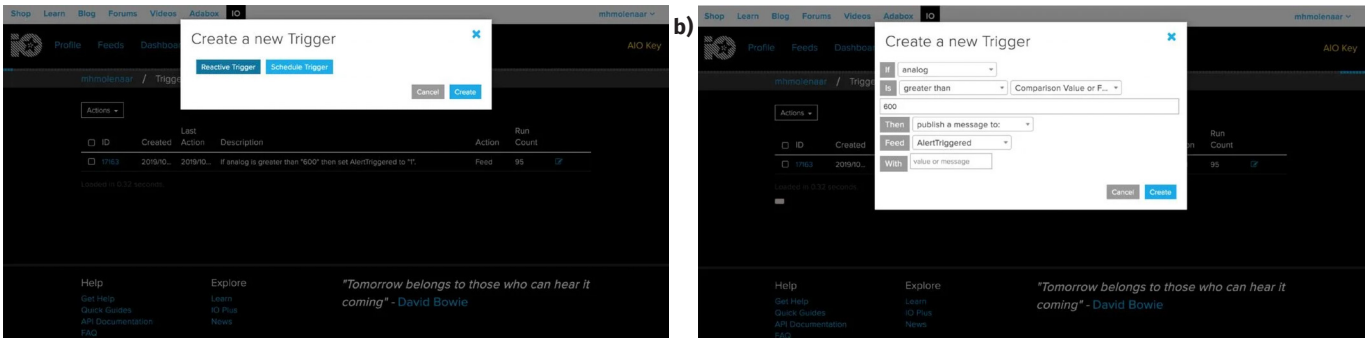
Po wprowadzeniu tych zmian, możemy przetestować szkic. Wystarczy go skompilować i wgrać do naszego NodeMCU. Po uruchomieniu mikrokontrolera z nowym oprogramowaniem, system powinien połączyć się z Wi-Fi, a następnie nawiązać połączenie z Adafruit IO. Możemy teraz napiąć nasze mięśnie i sprawdzić, jak zachowuje się system. Z jednej strony zwraca on wartość pomiaru poprzez port szeregowy (prędkość transmisji 115200 bodów), jak pokazano na **rysunku 2c**. Taka sama wartość powinna być zaprezentowana w sieci na *dashboardzie* Adafruit IO, jak widzimy na **rysunku 2d**.

Krok 4: Konfiguracja wyzwalaczy w Adafruit IO

Do tej pory tylko wprowadzaliśm.in.ormacje do systemu Adafruit IO. Teraz zaczniemy także z systemu czytać. Skonfigurujemy system tak, że po wykryciu przekroczenia wartości powyżej ustalonego progu czujnika MyoWare, pasek ledowy RGB wykona predefiniowaną akcję. W tym celu musimy skonfigurować wyzwalacze (*trigger*) w Adafruit IO.

W pierwszej kolejności konfigurujemy dodatkowy kanał w Adafruit IO. Musimy utworzyć inny kanał, aby móc uzyskać dane wyjściowe. W tym celu tworzymy (analogicznie jak poprzednio) kanał o nazwie *AlertTriggered*. Teraz można utworzyć wyzwalacz. W menu Adafruit IO, obok nagłówków *Feed* i *Dashboard*, znajdziemy przycisk „Trigger", skorzystajmy z niego. Pozwoli to utworzyć nowy wyzwalacz. W otwartym w ten sposób menu (**rysunek 3a** oraz **3b**) najpierw wybieramy, czy ma być to wyzwalacz reaktywny, czy okresowy. Wybieramy reaktywny i podajemy wszystkie dane, jakie są wymagane (patrz rysunek 3b). W menu wyzwalacza musimy skonfigurować szereg parametrów:

- wybieramy kanał, w którym znajdują się dane analogowe z naszego sensora,
- jako wyzwalacz musimy wybrać komparator i uruchamiać go, gdy wartość przekroczy ustawiony próg,
- wprowadzamy bezpieczną wartość progu, na przykład 600,
- aby odebrać wiadomość w Arduino, musimy ustawić *„publish a message to"*,
- musimy wskazać kanał, w którym znajdzie się wyzwalacz, wybieramy utworzony wcześniej *AlertTriggered*.



Rysunek 3. Konfiguracja wyzwalaczy w Adafruit IO

Teraz możemy nacisnąć „utwórz”, aby wygenerować wyzwalacz.

Krok 5: Sterowanie diodami LED RGB

Aby sterować diodami Neopixel, musimy nasłuchiwać kanału *Alert-Triggered*, który przed chwilą utworzyliśmy. NodeMCU będzie sterowało paskiem LED-owym, który potrzebuje jednej linii cyfrowej i zasilania 5 V, w związku z czym podłączamy go w następujący sposób:

1. Lewe pole paska LED do zasilania, najlepiej 5 V, ale powinny one działać z zasilaniem 3,3 V z NodeMCU.
2. Środkowe pole – DIN (wejście danych) podłączamy do portu D5 NodeMCU.
3. Prawe pole paska LED podłączamy do masy.

Do sterowania systemem zmodyfikowano szkic z przykładu. Pełny kod programu pokazany został na **listingu 1**. Kod ten współpracuje z plikiem nagłówkowym *config.h*, w którym znajdują się dane konfiguracyjne dla programu. Po skompilowaniu i załadowaniu szkicu do NodeMCU możemy uruchomić system. Połączy się z Adafruit i zacznie pracę. Po pięciokrotnym przekroczeniu progu pomiaru z sensora (ustaliliśmy wartość 600) uruchomi się alarm sygnalizowany przez diody na pasku LED-owym Neopixel.

Nikodem Czechowski, EP

Źródło:

1. ***http://bit.ly/32Psh1m***

Listing 1.

```
/************************* Konfiguracja ************************/

#include „config.h”
#include „Adafruit_NeoPixel.h”

/************** Kod programu zaczyna się tutaj *****************/

//definicje dla paska LED
#define PIXEL_PIN              D5
#define PIXEL_COUNT  10
#define PIXEL_TYPE   NEO_GRB + NEO_KHZ800

//pin analogowy dla sensora
#define MUSCLESENSOR_PIN A0

//Uruchamia alert po X przekroczeniu wartości w wyzwalaczu
#define ALERT_AFTER_TIMES 6

//Stan sensora MyoWare
int current = 0;
int last = -1;

//Stan alarmu
bool alertActive = false;
int alertLevel = 0;
int secondsPassed = 0;

// Neopixel
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(
PIXEL_COUNT, PIXEL_PIN, PIXEL_TYPE);

//Podłączenie do kanału pod nazwą „analog”
AdafruitIO_Feed *analog = io.feed(„analog”);

//Podłączenie do wyzwalacza, który wysyła wartość 1,
//gdy zostanie uruchomiony
AdafruitIO_Feed *alertTriggered = io.feed(„AlertTriggered”);


void setup() {

    //Uruchomienie konsoli szeregowej
    Serial.begin(115200);

    //oczekiwanie, aż konsola uruchomi się
    while(! Serial);

    //połączenie z  io.adafruit.com
    Serial.print(„Connecting to Adafruit IO”);
    io.connect();

    // obsługa wyzwalacza przychodzącego
    alertTriggered->onMessage(handleTrigger);

    //oczekiwanie na połączenie z Adafruit IO
    while(io.status() < AIO_CONNECTED) {
        Serial.print(„.”);
        delay(500);
    }

    //połączenie
    Serial.println();
    Serial.println(io.statusText());

    //inicjalizacjia Neopixel
    pixels.begin();
    setToWhite();
}

void loop() {

    //io.run(); jest wymagane przez wszystkie szkice
    //powinno być zawsze na starcie głównej pętli programu
    //służy do połączenia z io.adafruit.com.
    io.run();

    //aktualny stan sensora
    current = analogRead(MUSCLESENSOR_PIN);

    //wyjście, jeśli wartość się nie zmieniła
    if(current == last)
        return;

    //zapisanie obecnej wartości zmierzonej
    //z sensora do kanału analogowego
    Serial.print(„sending -> „);
    Serial.println(current);
    analog->save(current);

    // zapisujemy wartość sensora
    last = current;

    // poczekaj dwie sekundy
    delay(2000);

    testIfShouldReset();
}

// obsługa wyzwalacza
void handleTrigger(AdafruitIO_Data *data) {
    if(data->toInt() == 1) {
        alertLevel++;
        Serial.println(„Alert level up:”);
        Serial.println(alertLevel);

        //Uruchomienie alarmu w momencie, gdy wyzwalacza
        //zgłosił się więcej niż ALERT_AFTER_TIMES
        if(alertLevel >= ALERT_AFTER_TIMES) {
            alertOn();
        }
    }
}

//Uruchom akarm, wysyłając informacje do diod LED
void alertOn() {
    alertActive = true;
    setToRed();
    Serial.println(„Alert on”);
}

// Wyłącz alarm
void alertOff() {
    //ściemnianie diod LED poprzez obniżanie jasnoości,
    //jaka jest ustawiona
    for(int brightness = 255; brightness > 0; brightness -= 5) {
        pixels.setBrightness(brightness);
        pixels.show();
        delay(100);
    }

    //Reset alert variables
    alertLevel = 0;
    alertActive = false;
    Serial.println(„Alert off”);

    setToWhite();
}

//Zapalenie diod na biało
void setToWhite() {
    for (int i = 0; i < PIXEL_COUNT; ++i) {
        pixels.setPixelColor(i, 255, 255, 255);
    }
    pixels.setBrightness(255);
    pixels.show();
}

//Zapalenie diod na czerwono
void setToRed() {
    for (int i = 0; i < PIXEL_COUNT; ++i) {
        pixels.setPixelColor(i, 255, 0, 0);
    }
    pixels.setBrightness(255);
    pixels.show();
}

//Wyłączenie alarmu po jednej minucie
void testIfShouldReset() {
    if(alertActive) {
        secondsPassed += 2;
        // 60 to liczba sekund, możemy to zmienić
        if(secondsPassed >= 60) {
            alertOff();
        }
        //dane o liczbie sekund, jaka upłynęła od alarmu,
        //przesyłana jest przez port szeregowy
        Serial.println(„Seconds passed:”);
        Serial.println(secondsPassed);
    } else {
        secondsPassed = 0;
    }
}
```