



Mikrokontrolery dla hobbystów

Projekty DIY w języku C i C++

Miguel Angel Garcia-Ruiz
Pedro Cesar Santana Mancilla

Helion



Tytuł oryginału: DIY Microcontroller Projects for Hobbyists: The ultimate project-based guide to building real-world embedded applications in C and C++ programming

Tłumaczenie: Marcin Machnik

ISBN: 978-83-283-8947-2

Copyright © Packt Publishing 2021. First published in the English language under the title 'DIY Microcontroller Projects for Hobbyists' – (9781800564138).

Polish edition copyright © 2022 by Helion S.A.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/mikhob>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

O autorach	11
O recenzentach	12
Przedmowa	13
Rozdział 1. Wprowadzenie do mikrokontrolerów i płytek z mikrokontrolerami	19
Wymogi techniczne	20
Wprowadzenie do mikrokontrolerów	21
Płytki z mikrokontrolerami	22
Przegląd wykorzystywanych w książce analogowych i cyfrowych elementów elektronicznych	23
Podstawowe elementy elektryczne	24
Dioda	26
Płytki prototypowa	28
Opis płytek Blue Pill i Curiosity Nano	29
Instalowanie IDE	31
Twój pierwszy projekt — migająca dioda LED	37
Uruchomienie kodu z migającą diodą na płytce Blue Pill	38
Uruchomienie kodu z migającą diodą na płytce Curiosity Nano	40
Podsumowanie	42
Dalsza lektura	42
Rozdział 2. Instalacja środowisk programistycznych oraz programowanie mikrokontrolerów w C	43
Wymogi techniczne	44
Wprowadzenie do programowania w C	44
Podstawowa struktura języka C	45
Wstęp do programowania mikrokontrolera Curiosity Nano	57

Wstęp do programowania mikrokontrolera Blue Pill	58
Przykład — programowanie i wykorzystywanie wbudowanej w płytke mikrokontrolera diody LED	61
Programowanie diody na płytce Blue Pill	61
Podsumowanie	64
Dalsza lektura	64
Rozdział 3. Włączanie i wyłączanie diody przyciskiem	65
Wymogi techniczne	66
Przyciski	66
Szum elektryczny przycisków	68
Debouncing sprzętowy	69
Debouncing softwarowy	70
Podłączenie diody do płytki mikrokontrolera z wykorzystaniem wbudowanego rezystora podciągającego	72
Softwarowy debouncing przycisku podłączonego do Blue Pill	73
Włączanie i wyłączanie diody przyciskiem podłączonym do płytki Curiosity Nano	76
Testowanie przycisku	78
Podsumowanie	79
Dalsza lektura	79
Rozdział 4. Pomiar ilości światła za pomocą fotorezystora	81
Wymogi techniczne	81
Czujniki	82
Definicja czujnika	82
Czym są moduły z czujnikami?	83
Fotorezystor	84
Podłączanie fotorezystora do portu płytki mikrokontrolera	86
Podłączenie fotorezystora do płytki Blue Pill	86
Podłączenie fotorezystora do płytki Curiosity Nano	89
Podłączenie modułu z fotorezystorem do płytek z mikrokontrolerem	92
Testowanie fotorezystora	95
Podsumowanie	96
Dalsza lektura	96
Rozdział 5. Pomiar temperatury i wilgotności	97
Wymogi techniczne	97
Czujnik wilgotności i temperatury DHT11	98
Czym jest wilgotność względna?	98
Piny DHT11 i ich opis	99
Podłączenie modułu z czujnikiem DHT11 do płytki z mikrokontrolerem	100
Podłączenie DHT11 do płytki Blue Pill	101
Podłączenie wyświetlacza LCD do Blue Pill	103
Podłączenie czujnika temperatury LM35 do płytki Curiosity Nano	106
Programowanie odbierania danych z czujnika	108
Programowanie współpracy czujnika DHT11 z Blue Pill	108
Programowanie czujnika i wyświetlacza LCD	110
Programowanie współpracy czujnika LM35 z Curiosity Nano	111

Wyświetlanie danych o wilgotności i temperaturze w oknie monitora portu szeregowego	112
Tworzenie wykresów	113
Podsumowanie	114
Dalsza lektura	114
Rozdział 6. Alarm świetlny SOS kodem Morse'a na jasnej diodzie LED	115
Wymogi techniczne	116
Kod Morse'a i sygnał SOS	116
Superjasne diody LED i obliczanie niezbędnego rezystora	118
Podłączenie rezystora i superjasnej diody LED do Blue Pill	120
Podłączenie superjasnej diody LED do Curiosity Nano	123
Programowanie sygnału SOS	126
Sygnał SOS na Curiosity Nano	128
Testowanie alarmu świetlnego	129
Podsumowanie	130
Dalsza lektura	130
Rozdział 7. Przełącznik akustyczny	132
Wymogi techniczne	132
Podłączenie mikrofonu do portu płytki z mikrokontrolerem	134
Moduł z mikrofonem elektretowym	134
Podłączenie elementów	135
Programowanie przełącznika akustycznego	139
Programowanie przełącznika na dwa kłaśnięcia	141
Programowanie przełącznika akustycznego z timerem między kłaśnięciami	143
Ulepszanie działania projektu	145
Podsumowanie	146
Dalsza lektura	146
Rozdział 8. Czujnik gazów	147
Wymogi techniczne	147
Czujnik gazów MQ-2	148
Podłączenie czujnika MQ-2 do płytki z mikrokontrolerem	149
Podłączenie na potrzeby odczytu cyfrowego	149
Podłączenie na potrzeby odczytu analogowego	153
Kod odczytujący stężenie gazów z modułu czujnika	154
Kod do odczytu cyfrowego	155
Kod do odczytu analogowego	157
Test systemu	158
Podsumowanie	163
Dalsza lektura	163
Rozdział 9. IoT — system rejestrujący temperaturę	164
Wymogi techniczne	165
Podłączenie czujnika temperatury do płytki Blue Pill	165
Czujnik temperatury DS18B20	166
Podłączenie komponentów	167

Programowanie odczytu temperatury	170
Podłączenie modułu ESP8266	174
Moduł Wi-Fi ESP8266	174
Podłączenie modułu Wi-Fi ESP8266	175
Programowanie wysyłania odczytów temperatury do sieci	178
Podłączenie płytki Blue Pill do sieci	181
Podsumowanie	183
Dalsza lektura	183
Rozdział 10. IoT — czujnik nawilżenia rośliny	184
Wymogi techniczne	185
Podłączenie czujnika wilgotności gleby do płytki Blue Pill	185
Czujnik wilgotności gleby	185
Podłączenie elementów	186
Odczyt danych z modułu czujnika wilgotności gleby	189
Programowanie wysyłania odebranych danych do sieci	191
Wyświetlanie danych z czujnika przez sieć Wi-Fi	198
Podsumowanie	200
Dalsza lektura	201
Rozdział 11. IoT — pomiar energii słonecznej (napięcia)	202
Wymogi techniczne	203
Podłączenie ogniwa fotowoltaicznego do płytki Blue Pill	203
Ogniwo fotowoltaiczne	203
Czujnik napięcia B25	205
Podłączenie elementów	206
Odczyt danych z czujnika napięcia	209
Programowanie wysyłania zmierzonych danych do internetu	211
Prezentowanie danych z czujnika w internecie	214
Podsumowanie	219
Dalsza lektura	219
Rozdział 12. Cyfrowy pomiar temperatury ciała	220
Wymogi techniczne	221
Programowanie komunikacji I2C	221
Protokół I2C	221
Programowanie I2C	223
Podłączenie czujnika na podczerwień do płytki z mikrokontrolerem	226
Czujnik MLX90614	226
Moduł GY-906	228
Podłączenie czujnika do Arduino Uno	229
Podłączenie Arduino Uno z Blue Pill	231
Prezentacja temperatury na wyświetlaczu LCD	232
Test termometru	234
Podsumowanie	235
Dalsza lektura	236

Rozdział 13. Alarm dystansu społecznego	237
Wymogi techniczne	238
Programowanie brzęczyka piezoelektrycznego	239
Podłączenie komponentów	240
Podłączenie czujnika ultradźwiękowego do płytki z mikrokontrolerem	242
Podłączenie komponentów	243
Programowanie odbierania danych z czujnika ultradźwiękowego	244
Test pomiaru odległości	246
Podsumowanie	250
Dalsza lektura	250
Rozdział 14. Timer dwudziestosekundowego mycia rąk	251
Wymogi techniczne	252
Programowanie licznika czasu (timera)	252
Prezentacja licznika na wyświetlaczu	254
Podłączenie czujnika ultradźwiękowego do Blue Pill	256
Co to jest czujnik ultradźwiękowy?	257
Jak działa czujnik ultradźwiękowy?	257
Składamy wszystko razem — pomyśl o obudowie ochronnej!	261
Test timera	262
Podsumowanie	264
Dalsza lektura	264

Włączanie i wyłączanie diody przyciskiem

W tym rozdziale opiszemy i poćwiczymy włączanie i wyłączanie diody LED przyciskiem typu **tact switch** (przełącznikiem taktowym) podłączonym do płytki z mikrokontrolerem. Tego typu przycisk to przydatny komponent, który funkcjonuje jako przełącznik i jest używany do zamykania lub otwierania obwodu elektrycznego. Będziemy go stosować do inicjowania lub aktywowania procesu na płytce z mikrokontrolerem. Dane wejściowe przekazywane przez przycisk są istotne w wielu projektach z mikrokontrolerami, które wymagają udziału człowieka. Konkretnie zawartość rozdziału prezentuje się następująco:

- przedstawienie przycisków typu tact switch,
- wyjaśnienie kwestii zakłóceń elektrycznych przycisku,
- podłączenie do portu płytki mikrokontrolera diody LED z wykorzystaniem rezystora podciągającego,
- przetestowanie przycisku.

Z lektury tego rozdziału dowiesz się, jak podłączyć przycisk do płytek Curiosity Nano i Blue Pill oraz nauczysz się programować płytki w taki sposób, by dane z przycisku włączały lub wyłączały diodę. Poznasz także metody redukowania problemu **szumu elektrycznego** z przycisków. Przekonasz się, że próby rozwiązania tej kwestii nie są łatwe. W sekcji „Szum elektryczny przycisków” mówimy o tym, że nie wszystkie przyciski są w stu procentach wolne od błędów produkcyjnych i korzystanie z nich może indukować szumy elektryczne.

Wymogi techniczne

Narzędzia programistyczne, z jakich będziemy korzystać w tym rozdziale, to **MPLAB X IDE** oraz **Arduino IDE**. Kod użyty w tym rozdziale znajduje się w repozytorium tej książki w serwisie GitHub pod adresem:

<https://github.com/PacktPublishing/DIY-Microcontroller-Projects-for-Hobbyists/tree/master/Chapter03>.

Filmik dotyczący tego rozdziału znajdziesz pod adresem: <https://bit.ly/3cVfZLM>.

Przykładowe kody z tego rozdziału służą do włączania i wyłączania diody LED za pośrednictwem płytek Curiosity Nano i Blue Pill. Przewodnik na temat instalacji środowisk IDE i korzystania z nich znajdziesz w rozdziale pierwszym. W tym rozdziale będą nam także potrzebne następujące elementy elektroniczne:

- Płytki prototypowa.
- Płytki Blue Pill i Curiosity Nano.
- Kabel micro-USB do podłączenia płytek do komputera.
- Interfejs ST-Link/V2 do wgrania skompilowanego kodu na płytkę Blue Pill. Pamiętaj, że ST-Link/V2 wymaga czterech żeńsko-żeńskich przewodów Arduino ze zworkami.
- Jedna dioda LED. Jakiegokolwiek koloru. W ćwiczeniach zwykle korzystamy z czerwonej.
- Jeden rezystor 220 Ω o mocy 1/4 wata.
- Cztery męsko-męskie przewody Arduino ze zworkami do podłączenia rezystora i przycisku do płytek.
- Przycisk typu **tact switch**, normalnie otwarty (NO).

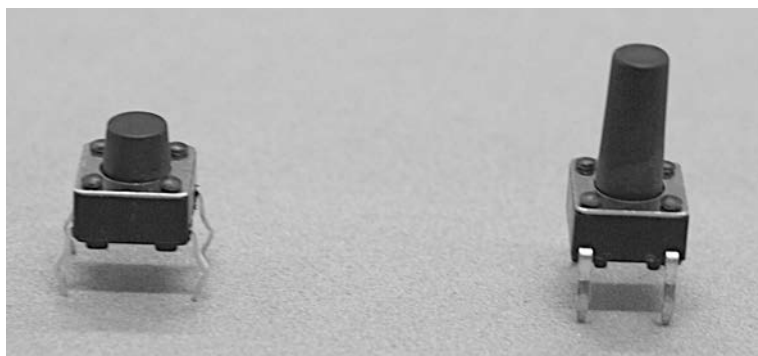
Następna sekcja zawiera zwięzłe wprowadzenie do przycisków używanych w obwodach elektrycznych.

Przyciski

Przycisk to urządzenie elektroniczne, które działa na zasadzie mechanicznego **przełącznika** i może służyć do zamknięcia lub otwarcia obwodu. Przyciski nazywa się także *chwilowymi* lub *impulsowymi*. Są one zbudowane z twardych materiałów w rodzaju plastiku i mają w środku małą metalową sprężynę, która odpowiada za zetknięcie dwóch przewodów lub styków, pozwalając na przepływ elektryczności po naciśnięciu przycisku (w przyciskach **normalnie otwartych**) lub po ich puszczeniu (w przyciskach **normalnie zamkniętych**). Po zwolnieniu przycisku sprężyna się wycofuje, obwód zostaje przerwany i przez styki przestaje płynąć prąd.

Przyciski przydają się do manualnego kontrolowania lub inicjowania procesów w obwodach elektrycznych, w tym takich z udziałem płytek z mikrokontrolerami. Poniższe zdjęcie przedstawia przycisk normalnie zamknięty (po lewej) i normalnie otwarty (po prawej).

Jak widać na rysunku 3.1, normalnie otwarty przycisk (ten po prawej) wygląda na wciśnięty. Zwróć uwagę, że nóżki podcina się do płytki z mikrokontrolerem.



Rysunek 3.1. Przyciski normalnie zamknięty (po lewej) i normalnie otwarty (po prawej)

Normalnie otwarte i normalnie zamknięte przyciski mogą wyglądać identycznie w zależności od producenta i modelu. Jeśli nie masz pewności, podłącz przycisk do płytki z mikrokontrolerem i sprawdź, z jakim rodzajem masz do czynienia. Jeśli przycisk wysłał sygnał logiczny bez naciskania, jest normalnie zamknięty. W tym rozdziale nauczysz się podłączać przycisk do płytki z mikrokontrolerem.

Typowym zastosowaniem przycisku w projektach z mikrokontrolerami jest podłączanie masy lub dodatniego napięcia do pinu wejściowego lub odłączanie masy lub napięcia od tego pinu. Mikrokontroler odczytuje taką zmianę napięcia na porcie I/O (pinie) i inicjuje jakiś proces.

Przyciski mogą być różnej wielkości. Duże i solidne przydają się w zastosowaniach przemysłowych, gdy operator musi móc je szybko zlokalizować i wcisnąć. Mniejsze zwykle stosuje się w urządzeniach elektrycznych, takich jak klawiatury komputera czy telefony stacjonarne. W tym rozdziale użyjemy małego przycisku, który znajdziesz w wielu zestawach elementów elektronicznych, w tym takich z mikrokontrolerami. Co ciekawe, płytki Blue Pill i Curiosity Nano także są wyposażone w taki przycisk. W obu przypadkach służy on do resetowania uruchomionego na nich programu. Istnieją dwa rodzaje przycisków: **normalnie otwarte** i **normalnie zamknięte**. Przyjrzyjmy się im bliżej.

- **Normalnie otwarte.** Gdy przycisk nie jest naciskany, styki są rozwarte, czyli nie zamykają obwodu. Zwierają się (zamykają obwód) przy każdym naciśnięciu przycisku. Po wciśnięciu przycisku obwód zostanie zamknięty. To najpopularniejszy rodzaj przycisku. Przydaje się do chwilowej aktywacji lub inicjalizacji procesu, na przykład zresetowania mikrokontrolera.

- **Normalnie zamknięte.** W domyślnym stanie ten przycisk zamyka obwód, co oznacza, że jego styki są zwarte bez konieczności naciskania. Rozłączamy styki (i otwieramy obwód, do którego przycisk jest podłączony) poprzez naciśnięcie. Ten przycisk przydaje się, gdy chcemy na chwilę wyłączyć lub przerwać obwód elektryczny. Na przykład przerwać połączenie czujnika z mikrokontrolerem, gdy z jakiegoś powodu chcemy przerwać napływ danych.

W następnej sekcji opisujemy problem dotyczący wielu czujników, a związany z **szumem elektrycznym**. Tego rodzaju szum jest czasem bardzo trudny (lecz nie niemożliwy) do zminimalizowania.

Szum elektryczny przycisków

Wiele przycisków generuje szumy. Mają one negatywny wpływ na funkcjonalność obwodu elektrycznego, do którego przycisk jest podłączony, oraz mogą wywoływać nieprzewidywalne reakcje mikrokontrolera.

Główny problem z przyciskami polega na tym, że *nie są idealne*. Nie zwierają styków w jednej chwili i w wielu przypadkach powstaje szum elektryczny. Dzieje się tak dlatego, że nie wszystkie przyciski są wolne od błędów produkcyjnych. Gdy podłączymy przycisk bezpośrednio do portu I/O mikrokontrolera, możemy odnieść wrażenie, że po naciśnięciu wszystko przebiega tak jak należy. Nam się wydaje, że naciskamy tylko raz, ale z punktu widzenia mikrokontrolera przycisk został wcisnięty wiele razy w bardzo krótkich odstępach czasu, co jest wynikiem wygenerowanego przez przycisk szumu elektrycznego. Szum elektryczny możemy zdefiniować jako przypadkowe impulsy elektryczne lub sygnały w obwodzie elektrycznym. Takie sygnały mają mocno zróżnicowane poziomy napięć — od bardzo niskich do bardzo wysokich — i mogą pojawiać się z przypadkową częstotliwością. Istnieje wiele źródeł szumów elektrycznych, między innymi ciepło, wadliwe komponenty, ruchy mechaniczne i luźne połączenia w obwodzie.

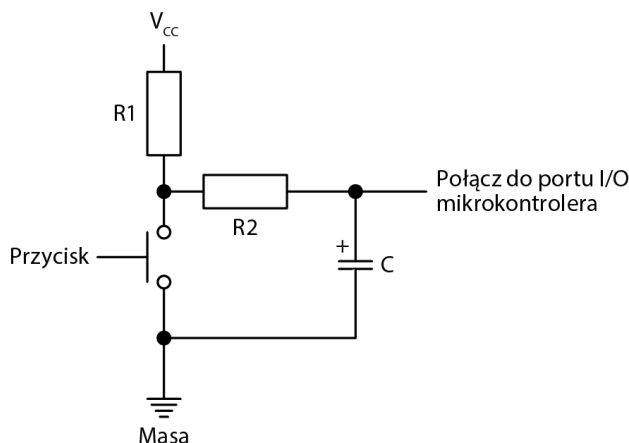
Niepożądany szum elektryczny z przycisków jest niemal zawsze wynikiem zjawiska zwanego **odbijaniem**, powodowanym przez tarcie i mechaniczne ruchy wewnętrznych metalowych części przycisku i sprężyny. Musimy usunąć małe pulsacje napięcia, czyli wykonać tzw. **debounce**, żeby zmniejszyć szum elektryczny i zamknąć obwód (gdy używamy przycisku normalnie otwartego) w czysty i efektywny sposób. Jeśli tego nie zrobimy, wewnętrzne styki mogą nieprawidłowo zamykać obwód przy każdym naciśnięciu, co wpłynie na funkcjonalność całego obwodu lub wejścia mikrokontrolera. Przycisk powinien wysyłać albo zero woltów (logiczny stan niski — LOW), albo 3,3V (logiczny stan wysoki — HIGH). Jeśli nie usuniemy małych pulsacji, powstały szum elektryczny może wpłynąć na te poziomy logiczne i mikrokontroler może przestać je poprawnie rozpoznawać.

Logiczny stan wysoki (HIGH) zarówno dla płytki Blue Pill, jak i Curiosity Nano to 3,3V. Pamiętaj, że dla niektórych mikrokontrolerów, na przykład z rodziny Arduino, HIGH to 5V, a nie 3,3V.

Istnieje kilka technik radzenia sobie z szumem elektrycznym przycisków. Przykładowo ten rodzaj szumu można znacznie zminimalizować za pomocą połączonych z przyciskiem dodatkowych elementów elektronicznych lub za pomocą kodu. Pokażemy te sposoby w następnych kilku sekcjach.

Debouncing sprzętowy

Jednym ze sposobów na zredukowanie szumu przycisku jest podłączenie do przycisku kondensatora i dwóch rezystorów (taki układ często nazywa się **filtrem RC** lub filtrem dolno-przepustowym), tak jak na poniższym schemacie z rysunku 3.2. Gdy wciśniemy przycisk, kondensator się naładuje. Gdy puścimy przycisk, kondensator przez krótki czas będzie naładowany, lecz szybko rozładuje się przez rezystor. Naładowany kondensator reprezentuje logiczny stan wysoki (HIGH) do odczytania przez mikrokontroler. Wszelkie chwilowe szумы, które wstąpią w stanie naładowania, zostaną zignorowane, ponieważ kondensator podaje wtedy logiczny stan wysoki (HIGH).



Rysunek 3.2. Przycisk z filtrem RC¹

Powyższy schemat (rysunek 3.2) zawiera dwa rezystory, **R1** i **R2**, normalnie otwarty przycisk i kondensator, **C**. Rezystory i kondensator tworzą filtr RC. Pamiętaj, że **V_{cc}** oznacza dodatnie napięcie, czyli 3,3V dla Curiosity Nano i Blue Pill. Generalnie napięcie uzyskuje się z jednego z pinów płytki opisanego jako 3,3V lub V_{cc}. Dodatkowo możesz podpiąć filtr RC do jednego z pinów płytki opisanego jako masa. Jak widać na powyższym schemacie z rysunku 3.2, te trzy elementy nadają się do zredukowania szumu elektrycznego przycisku. Typowe wartości dla **R1**, **R2** i **C** to odpowiednio 10 kΩ, 10 kΩ i 0.1 mikrofaraadów, chociaż czasem trzeba dobrać inne

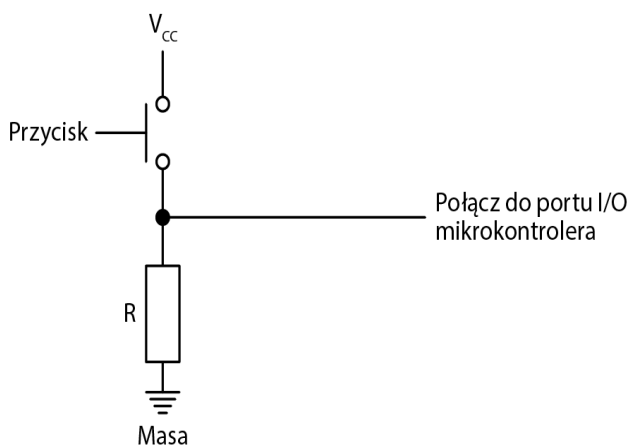
¹ Na schemacie zastosowano — zgodnie z oryginałem — symbol **V_{cc}** dla oznaczenia napięcia. Wprawdzie w Polsce stosuje się raczej zapis **U_{cc}**, jednak zdecydowałem się zastosować symbolikę z oryginału po to, by zachować zgodność z odpowiednim oznaczeniem na płytce. W dalszej części książki stosuję taką samą taktykę — *przyp. tłum.*

wartości, gdy filtr nie działa jak należy, gdyż szum elektryczny przycisku nie zawsze jest taki sam. Matematyczny wzór na obliczanie parametrów filtra RC szczegółowo wyjaśnia Jack Ganssle w raporcie *A guide to debouncing*.

Uwzględniliśmy w tej sekcji filtr RC na wypadek, gdyby metoda programistyczna okazała się w Twoim przypadku nieskuteczna. W następnej sekcji wyjaśniamy, jak wykonać debouncing *wyłącznie* softwarowy.

Debouncing softwarowy

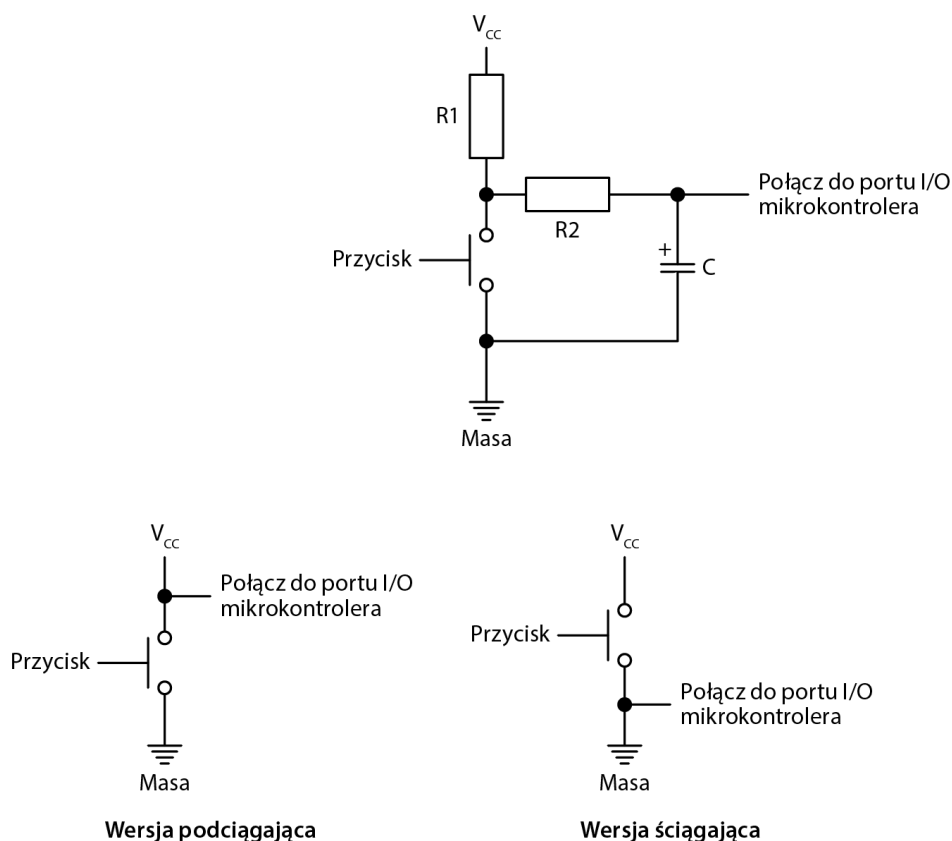
Fałszywe sygnały przycisku możemy też zminimalizować za pomocą kodu. Nasz trik w kodzie polega na tym, że nakażemy ignorowanie sygnałów o bardzo krótkim czasie trwania (w granicach kilkunastu milisekund) tuż po naciśnięciu przycisku podłączonego do naszego obwodu. Poniższy schemat (rysunek 3.3) ilustruje sposób podłączenia przycisku bezpośrednio do portu I/O mikrokontrolera, gdy debouncing jest dokonywany softwarowo.



Rysunek 3.3. Przycisk podłączony do portu I/O mikrokontrolera z rezystorem ściągającym

Powyższy schemat zawiera rezystor ściągający (**R**), który, gdy przycisk nie jest wciśnięty, wymusza obecność na wejściu mikrokontrolera zerowego napięcia (logicznego STANU NISKIEGO), gdyż jest podłączony do masy. Typowa oporność tego rezystora to 10 kΩ. Taki rezystor przydaje się, gdy chcemy, żeby na wejściu był stale stan LOW, który zmienia się na HIGH tylko, gdy naciskamy przycisk. Przydaje się to do wywoływania procesów, na przykład włączenia światła w obwodzie. Poniższy schemat (rysunek 3.4) pokazuje natomiast, jak podłączyć do portu mikrokontrolera rezystor podciągający, który wymusza na wejściu obecność napięcia 3,3V (V_{CC}).

Typowa wartość rezystora w powyższym przykładzie to także 10 kΩ.



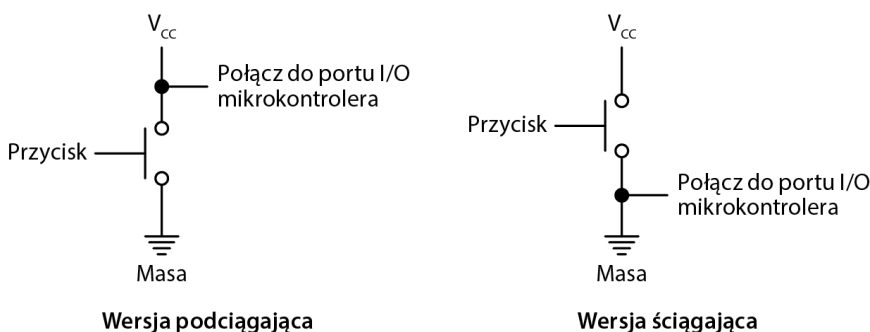
Rysunek 3.4. Przycisk podłączony do portu I/O mikrokontrolera z rezystorem podciągającym

Musisz podpiąć do portu mikrokontrolera albo rezystor ściągnięty, albo podciągający, bo gdy nic nie jest podłączone (a tak jest w sytuacji, gdy nie naciskamy przycisku), na porcie pojawi się niezdeterminowane (losowe) napięcie, wynikające z wewnętrznych połączeń w płytce. Takie napięcie nazywamy **plywającym**.

Poprzedni schemat zawiera przycisk normalnie otwarty. Po jego naciśnięciu napięcie wejściowe zmienia się na logiczny stan wysoki (HIGH), czyli 3,3V. Pamiętaj, że to napięcie może wynosić 5V w zależności od używanego mikrokontrolera. V_{CC} i masa są podłączone do płytki z mikrokontrolerem.

Na szczęście wiele mikrokontrolerów zawiera wbudowane rezystory podciągające i ściągnięte, które są podłączone do portów I/O i można je aktywować programowo. Blue Pill zawiera oba te rodzaje! Oznacza to, że możemy podpiąć przycisk bezpośrednio do portu I/O bez konieczności podłączania zewnętrznego rezystora, musimy jedynie aktywować w kodzie wbudowany rezystor podciągający lub ściągnięty. Przyspiesza to pracę nad prototypami wymagającymi wielu przycisków, ale nie zawsze jest to rozwiązanie idealne.

Poniższy schemat (rysunek 3.5) przedstawia dwa sposoby podpięcia przycisku bezpośrednio do portu I/O, który jest wyposażony w rezystor ściągający lub podciągający aktywowany softwarowo.



Rysunek 3.5. Przycisk podłączony bezpośrednio do portu I/O

Jak widzisz, w metodzie softwarowej nie ma potrzeby podłączania filtra RC. Sprawdza się ona w większości przypadków i pozwala oszczędzić czas i energię. Powinieneś jednak poeksperymentować i sprawdzić obie metody, jeśli nie potrafisz zniwelować szumu.

Każda metoda debouncingu ma swoje minusy. Metoda sprzętowa wymaga użycia z przyciskiem dodatkowych elementów elektronicznych. Te dodatkowe elementy trzeba też *kupić*. Metoda softwarowa, w której korzystamy z wbudowanych w mikrokontroler rezystorów podciągających lub ściągających, nie wymaga do podłączenia przycisku żadnych dodatkowych elementów, ale musimy wydłużyć kod o parę linijek zajmujących się kwestią debouncingu, a te instrukcje zajmą kilka cennych cykli przetworzeniowych mikrokontrolera. Mimo tego zalecamy tę drugą metodę dlatego, że jest łatwa w stosowaniu.

W następnej sekcji pokażemy przykłady podłączenia przycisku do płytek Blue Pill i Curiosity Nano z wykorzystaniem debouncingu softwarowego.

Podłączenie diody do płytki mikrokontrolera z wykorzystaniem wbudowanego rezystora podciągającego

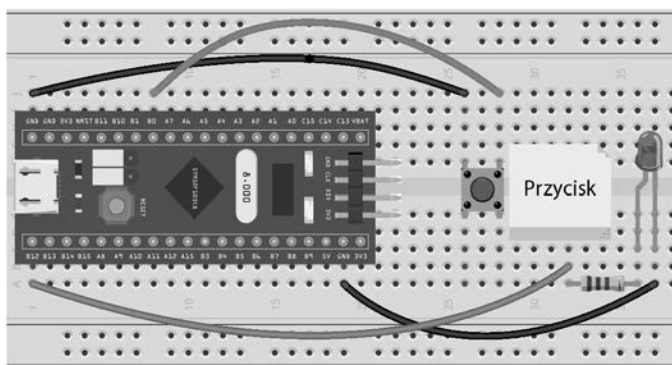
W tej sekcji nauczysz się podłączać przycisk zarówno do płytek Blue Pill, jak i Curiosity Nano. To proste ćwiczenie dla tych mikrokontrolerów demonstruje włączanie i wyłączanie diody LED poprzez podawanie przyciskiem na mikrokontroler logicznego STANU NISKIEGO. Jeśli chcemy w naszym przykładowym obwodzie użyć przycisku, musimy go podłączyć do portu wejściowego mikrokontrolera. Nie możemy zapomnieć o debouncingu, żeby uniknąć niepożądanych efektów.

W poniższej sekcji pokazujemy debouncing przycisku podpiętego do płytki Blue Pill za pomocą kodu. To najprostszy sposób na debouncing przycisku i możesz go wykorzystać w pozostałych rozdziałach tej książki.

Softwarowy debouncing przycisku podłączonego do Blue Pill

W tej sekcji pokażemy schemat połączeń, a następnie zdjęcie tych połączeń. Przyjrzymy się także kodowi, który demonstruje softwarowy debouncing przycisku.

Poniższy schemat (rysunek 3.6) przedstawia podłączenie przycisku bezpośrednio do portu I/O, który korzysta z wbudowanego rezystora podciągającego. Dioda LED i jej rezystor są podpięte do portu B12 płytki Blue Pill.



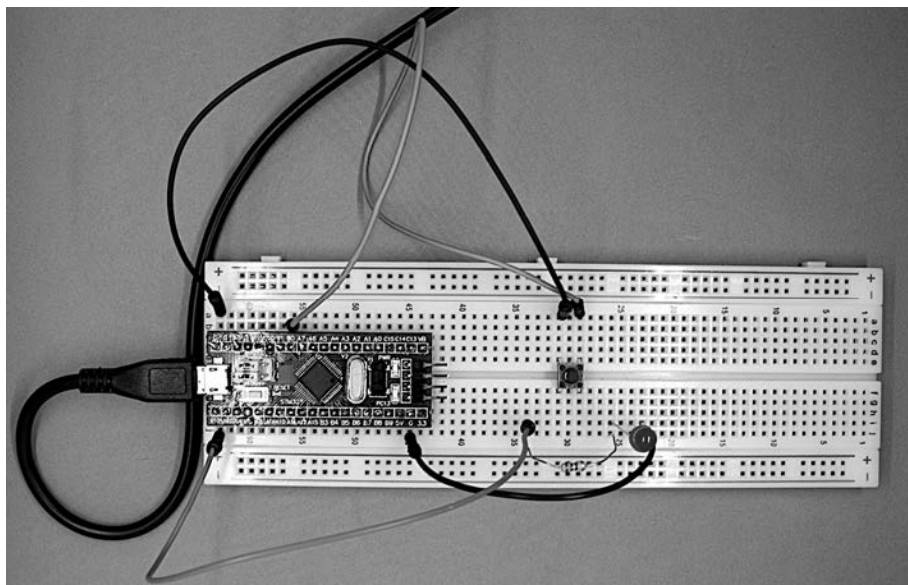
Rysunek 3.6. Podpięcie przycisku do płytki Blue Pill z wykorzystaniem jej wbudowanego rezystora podciągającego

Jak się przekonasz, dioda zaświeci się lub zgaśnie po każdym naciśnięciu przycisku.

Zwróć uwagę, że przycisk jest umieszczony na środku płytki prototypowej, na prawo od mikrokontrolera. Wykonaj poniższe kroki, żeby podłączyć go i diodę do płytki Blue Pill jak na powyższym schemacie:

1. Włóż płytkę Blue Pill do płytki prototypowej.
2. Włóż przycisk do płytki prototypowej i podłącz przewodem Arduino jeden z pinów do pinu uziemiającego płytki Blue Pill.
3. Podłącz przewodem Arduino drugi pin przycisku do portu B0 płytki Blue Pill.
4. Włóż do płytki prototypowej rezystor 220 Ω i przewodem Arduino podłącz jedną z jego nóżek do portu B12 płytki Blue Pill.
5. Włóż do płytki prototypowej diodę LED, aby anoda była połączona z drugą nóżką rezystora.
6. Podłącz przewodem Arduino katodę diody do pinu uziemiającego płytki Blue Pill.

Rysunek 3.7 pokazuje podłączenie przycisku do Blue Pill jak na wcześniejszym schemacie.



Rysunek 3.7. Podłączenie przycisku do płytki Blue Pill za pomocą przewodów Arduino

Jak widzisz, płytka Blue Pill ma porty uziemiające zarówno w górnym, jak i w dolnym rzędzie pinów. To ułatwia podłączanie komponentów.

Pamiętaj, że musisz podłączyć do płytki Blue Pill interfejs ST-Link/V2, żeby wgrać na nią kod za pomocą programu Arduino IDE, jak wyjaśniliśmy w rozdziale pierwszym.

Poniższy kod demonstruje softwarowy debouncing przycisku w mikrokontrolerze Blue Pill. Kod znajdziesz w repozytorium tej książki w serwisie GitHub wraz z dołączonymi komentarzami. Plik nosi nazwę *internal_pullup_debounce_Blue_Pill.ino*.

```
#define PinLED PB12
#define Pinprzycisku PB0
void setup() {
    pinMode(PinLED, OUTPUT);
    pinMode(Pinprzycisku, INPUT_PULLUP);
}
int odczyt_przycisku;
int stanLED = HIGH;
int stanprzycisku;
int ostatnistanprzycisku = LOW;
unsigned long ostatniczasdebouncingu = 0;
unsigned long opoznieniedebouncingu = 50;

void loop() {

    odczyt_przycisku=digitalRead(Pinprzycisku);
```

```

if (odczyt_przycisku != ostatnistanprzycisku) {
    ostatniczasdebouncingu = millis();
}
if ((millis() - ostatniczasdebouncingu) > opoznieniedebouncingu) {
    if (odczyt_przycisku != stanprzycisku) {
        stanprzycisku = odczyt_przycisku;
        if (stanprzycisku == HIGH) {
            stanLED = !stanLED;
        }
    }
}
digitalWrite(PinLED, stanLED);
ostatnistanprzycisku = odczyt_przycisku;
}

```

Powyższy kod odczeka 50 milisekund po naciśnięciu przycisku i dopiero wtedy zmienia stan LED. Ta wartość jest eksperymentalna, więc zmień ją, jeśli Twój przycisk zachowuje się niewłaściwie.

W środowisku Arduino IDE porty I/O płytki Blue Pill oznacza się literą P. Na przykład port B12 to PB12. Prócz tego etykiety portów (nazwy) muszą być wpisywane wielkimi literami.

Jak widać, powyższy kod nieprzerwanie odczytuje port B0 płytki Blue Pill. Po naciśnięciu przycisku port zostaje zwarty do masy. Wtedy port wyjściowy B12 wysyła logiczny stan wysoki (HIGH) i podłączona do tego portu dioda LED się zapala. Zanim naciśniesz przycisk, port B12 wysyła logiczny stan niski (LOW).

Niemal identycznie wygląda softwarowy debouncing przycisku w mikrokontrolerach Arduino. Opisana przez nas w tym rozdziale metoda bazuje nawet na metodzie dla płytek Arduino opisaną pod adresem:

<https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce>.

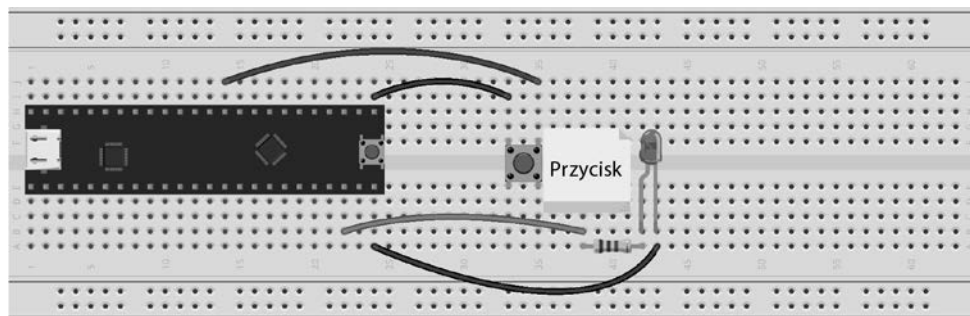
Jeśli Twoja dioda zapala się i gaśnie po naciśnięciu przycisku, gratulacje! Zwróć uwagę na to, jak mikrokontroler reaguje na naciśnięcie przycisku. Jeśli po naciśnięciu przycisku dioda kilka razy zachowa się niewłaściwie, spróbuj zmienić wartości rezystora lub kondensatora w przypadku debouncingu RC lub zmień wyrażone w milisekundach opóźnienie w przypadku debouncingu softwarowego.

W następnej sekcji opisujemy debouncing softwarowy przycisku podłączonego do płytki Curiosity Nano.

Włączanie i wyłączanie diody przyciskiem podłączonym do płytki Curiosity Nano

W tej sekcji dokonamy debouncingu softwarowego polegającego na odczekaniu zadanej liczby milisekund po naciśnięciu przycisku na mikrokontrolerze Curiosity Nano. Użyjemy do tego funkcji `__delay_ms()`. Pamiętaj, że przed tą funkcją trzeba wpisać dwa podkreślenia (`__`).

Poniższy schemat (rysunek 3.8) przedstawia podłączenie przycisku do płytki Curiosity Nano.



fritzing

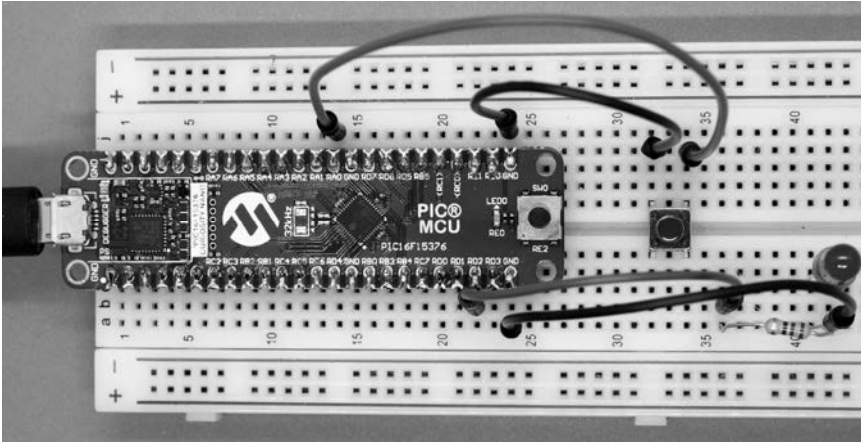
Rysunek 3.8. Przycisk podłączony bezpośrednio do płytki Curiosity Nano

Zwróć uwagę, że przycisk jest włożony na środku płytki po prawej stronie od mikrokontrolera.

Aby dokonać podłączenia zgodnie z powyższym schematem, wykonaj następujące kroki:

1. Włóż płytkę Curiosity Nano do płytki prototypowej.
2. Włóż przycisk do płytki prototypowej i podłącz przewodem Arduino jeden z jego pinów do pinu płytki Curiosity Nano.
3. Przewodem Arduino podłącz drugi pin przycisku do portu RA0 płytki Curiosity Nano.
4. Włóż do płytki prototypowej rezystor 220 Ω i przewodem Arduino połącz jedną z jego nóżek z portem RD2 płytki Curiosity Nano.
5. Włóż do płytki prototypowej diodę LED tak, by jej anoda była połączona z drugą nóżką rezystora.
6. Przewodem Arduino połącz katodę diody z pinem uziemiającym płytki Curiosity Nano.

Na rysunku 3.9 pokazaliśmy, jak to wygląda po połączeniu.



Rysunek 3.9. Płytką Curiosity Nano z podłączonym przyciskiem

Na powyższym zdjęciu widać, że płytką Curiosity Nano ma pin uziemiający (GND) zarówno w górnym, jak i w dolnym rzędzie pinów. To ułatwia nam podłączenie elementów w obwodzie.

Stworzyliśmy projekt dla środowiska MPLAB X IDE, który znajdziesz w repozytorium tej książki w serwisie GitHub. Projekt zawiera komentarze wyjaśniające poszczególne linie kodu. Aby otworzyć go w środowisku MPLAB X IDE, musisz go najpierw rozpakować. Plik nazywa się *16F15376_Curiosity_Nano_pushbutton.X.zip*.

Poniższy kod z tego projektu demonstruje sposób realizacji debouncingu softwarowego.

```
#include <xc.h>
#include <stdio.h>
#include "mcc_generated_files/mcc.h"
int odczyt_przycisku=0;
void main(void)
{
    SYSTEM_Initialize();
    IO_RD2_SetDigitalOutput();
    IO_RA0_SetDigitalInput();
    IO_RA0_SetPullup();
    IO_RD2_SetLow();
    while (1)
    {
        odczyt_przycisku=IO_RA0_GetValue();
        __delay_ms(100);
        odczyt_przycisku=IO_RA0_GetValue();
        if (odczyt_przycisku==LOW){
            IO_RD2_Toggle();
        }
    }
}
```

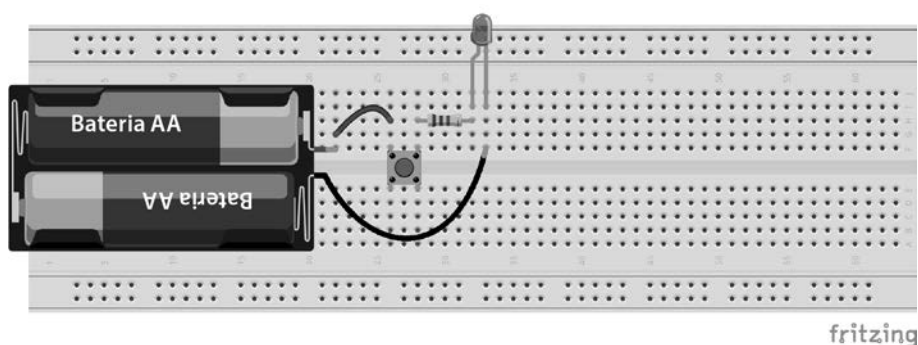
Powyższy kod odczytuje stan przycisku i odczekuje 100 milisekund, po czym odczytuje ponownie, aby sprawdzić, czy przycisk jest nadal wciśnięty. Znaleźliśmy tę wartość eksperymentalnie i zdaje się sprawdzać w większości przypadków. Pamiętaj, że może zająć konieczność jej zmodyfikowania w zależności od zachowania Twojego przycisku w obwodzie.

To podejście jest nieco inne niż zastosowane na płytce Blue Pill. Zaprogramowaliśmy czas oczekania, żeby zignorować szumy elektryczne, jakie mogą się wtedy pojawić. Jeśli dioda zapala się i gaśnie, gdy naciskasz przycisk, gratulacje! Potrafisz podłączyć i zaprogramować przycisk w obwodzie elektrycznym z mikrokontrolerem. Pamiętaj, że taki przycisk może służyć do inicjacji procesu lub aktywności w mikrokontrolerze.

W następnej sekcji wyjaśniamy, jak sprawdzić poprawność działania przycisku oraz czy mamy do czynienia z przyciskiem normalnie otwartym czy normalnie zamkniętym.

Testowanie przycisku

W tej sekcji skupimy się na testowaniu przycisku. Zanim podłączysz go do mikrokontrolera, dobrze jest sprawdzić poprawność mechanicznego działania. Ten test pozwala także ustalić, czy Twój przycisk jest normalnie zamknięty, czy normalnie otwarty. Rysunek 3.10 pokazuje, jak połączyć wszystkie komponenty potrzebne do wykonania testu.



Rysunek 3.10. Podłączenie przycisku do diody i koszyka z bateriami

Jak widać, do sprawdzenia przycisku nie potrzebujemy mikrokontrolera. Aby połączyć wszystkie potrzebne elementy i przetestować przycisk, wykonaj poniższe kroki:

1. Podłącz plus koszyka z bateriami do jednej z nóżek przycisku.
2. Drugą nóżkę przycisku połącz z nóżką rezystora 220 Ω .
3. Połącz rezystor z anodą diody LED.
4. Połącz katodę diody z minusem koszyka z bateriami. Uważaj przy podłączaniu diody, bo wpięta na odwrót się nie zaświeci.

5. Po podłączeniu wszystkiego sprawdź, czy dioda świeci się bez naciskania przycisku. Jeśli tak, masz przycisk normalnie zamknięty. W takiej sytuacji po naciśnięciu przycisku powinna zgasnąć. Jeśli natomiast zapala się dopiero po naciśnięciu, to masz przycisk normalnie otwarty.
6. Naciśnij przycisk kilka razy. Jeśli dioda będzie się zachowywać niewłaściwie lub w ogóle się nie zapali, przycisk może być uszkodzony i wymagać będzie zastąpienia nowym albo baterie dają niewystarczające napięcie.

Podłączenie przycisku do diody i baterii powinno wystarczyć do sprawdzenia, czy przycisk jest sprawny.

Podsumowanie

W tym rozdziale dowiedzieliśmy się, jak działa przycisk i jak możemy zredukować problem szumu elektrycznego generowanego przez wiele przycisków za pomocą debouncingu. Można go zrealizować albo sprzętowo, albo softwarowo. Zwróciliśmy uwagę na istotność przycisków w wielu projektach elektronicznych wymagających interwencji człowieka — na przykład gdy chcemy manualnie zrestartować mikrokontroler, naciskając wbudowany przycisk. Umiejętność wykorzystywania przycisków w obwodach elektronicznych z mikrokontrolerami jest istotna, gdyż programujemy interakcję użytkownika, żeby umożliwić mu wszczynanie jakiegoś procesu na płycie mikrokontrolera.

W następnym rozdziale skupimy się na podłączeniu fotorezystora (czujnika mierzącego ilość światła w otoczeniu).

Dalsza lektura

- Ganssle J. G., *A guide to debouncing*, raport techniczny, The Ganssle Group, Baltimore, MD 2008.
- Gay W., *Beginning STM32: Developing with FreeRTOS, libopencm3, and GCC*, Apress, St. Catharines 2018.
- Horowitz P., Hill W., *The Art of Electronics*, [wyd. 3.], New York NY, Cambridge University Press 2015. Polskie tłumaczenie: Horowitz P., Hill W., *Sztuka elektroniki*, cz.1 i cz.2, tłum. Grażyna Kalinowska, Bogusław Kalinowski, Wydawnictwo Komunikacji i Łączności, Warszawa 2018.
- Microchip, *PIC16F15376 Curiosity Nano Hardware User Guide*, Microchip Technology, Inc., 2019, dostępne pod adresem: <http://ww1.microchip.com/downloads/en/DeviceDoc/50002900B.pdf>.
- Mims F.M., *Getting Started in Electronics*, Master Publishing, Inc, Lincolnwood, IL 2000.

- Ostapiuk R., Tay I., *Fundamentals of the C programming language*, Microchip Technology, Inc., 2020, dostępne pod adresem: <https://microchipdeveloper.com/tls2101:start>.
- Ward H.H., *C programming for the PIC microcontroller*, Apress, Nowy Jork, NY 2020.

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion

Mikrokontrolery są nieodłącznymi elementami setek przedmiotów: urządzeń RTV i AGD, maszyn, narzędzi, a nawet zabawek. Umożliwiają sterowanie ich działaniem i pozwalają na wykonanie określonych czynności. Umiejętność programowania mikrokontrolerów jest wysoce pożądana na rynku pracy i daje szerokie perspektywy zawodowe, a nauczyć się jej można dzięki tej książce. Ten wyjątkowo przystępny podręcznik przybliży od podstaw programowanie mikrokontrolerów i wprowadza w arkana elektroniki cyfrowej. Można go polecić zarówno doświadczonym programistom i robotykom, jak i początkującym hobbystom, którzy chcą się dowiedzieć, jak stosować języki C i C++ w programowaniu mikrokontrolera.

To przewodnik po standardach STM32 i PIC, które należą do najpopularniejszych i najczęściej używanych. Zawiera wiele czytelnych objaśnień, przykładów i gotowych programów. Autorzy przystępnie tłumaczą, jak pisać aplikacje sterujące mikrokontrolerami, a następnie prawidłowo je programować za pomocą narzędzi, które także zostały tu dokładnie omówione.

Dzięki książce:

- przyswoisz podstawową wiedzę na temat elektroniki cyfrowej
- przygotujesz środowisko i narzędzia niezbędne do programowania mikrokontrolerów
- nauczysz się używać w tym celu języków C i C++
- dowiesz się, jak stworzyć prosty program do sterowania mikrokontrolerem

Miguel Angel Garcia-Ruiz — profesor Algoma University w Kanadzie, specjalizuje się w robotyce i programowaniu mikrokontrolerów, jest autorem licznych prac naukowych opisujących ich zastosowanie. Tytuł doktora uzyskał na Sussex University w Anglii w ramach specjalizacji *computer science and artificial intelligence*.

Pedro Cesar Santana Mancilla — profesor Universidad de Colima w Meksyku, aktualnie CEO Asociación Mexicana de Interacción Humano Computadora (AMeIHC) i członek zarządu Mexicano de ACM SIGCHI (CHI-México). Specjalista w zakresie technologii informacyjno-komunikacyjnych, internetu rzeczy i programowania mikrokontrolerów.

Helion 	KOD KORZYŚCI Ściągnij po więcej!  
 helion.pl  HELION SA ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	ISBN 978-83-283-8947-2  9 788328 389472
Cena: 69,00 zł	

Packt