WYJŚCIE ANALOGOWE (PWM)

```
analogWrite(RedPin, 100);
```

Funkcja **analogWrite()** wyprowadza na pin napięcie od 0 do 5 V. Funkcja dzieli zakres od 0 do 5 V na 255 małych kroków. Należy pamiętać, że w tym kodzie nie włączamy diody LED do pełnej jasności (255), aby światło nocne nie było zbyt jasne. Zmień te wartości i zobacz, co się stanie.

ZAGNIEŻDŻONE WARUNKI

```
if(logic statement){
  if(logic statement){

  } }
```

Zagnieżdżona instrukcja **if** to jedna lub więcej instrukcji **if** „zagnieżdżonych" w innej instrukcji **if**. Jeśli nadrzędna instrukcja **if** jest prawdziwa, kod sprawdza każdą zagnieżdżoną instrukcję **if** i wykonuje każdą z nich, która jest prawdziwa. Jeśli nadrzędna instrukcja **if** jest fałszywa, żadna z zagnieżdżonych instrukcji nie zostanie wykonana.

WIĘCEJ OPERATORÓW LOGICZNYCH

```
(potentiometer > 0 &&

  potentiometer <= 150)
```

Te instrukcje **if** sprawdzają dwa warunki za pomocą operatora AND **&&**. W tej linii instrukcja **if** będzie prawdziwa tylko wtedy, gdy wartość zmiennej **potentiometer** jest większa niż 0 **ORAZ** jeśli wartość jest mniejsza lub równa 150. Używając **&&**, program pozwala diodzie LED mieć wiele stanów kolorów.

DEFINIACJA FUNKCJI

```
void function_name(){ }
```

To jest definicja prostej funkcji. Kiedy programiści chcą wielokrotnie używać wielu linii kodu, piszą funkcję. Kod w nawiasach klamrowych „wykonuje się" za każdym razem, gdy funkcja jest „wywoływana" w programie głównym. Każdy z kolorów diody LED RGB jest zdefiniowany w funkcji.

WYWOŁANIE FUNKCJI

```
function_name();
```

Ta linia „wywołuje" utworzoną funkcję. W dalszej części dowiesz się, jak tworzyć bardziej skomplikowane funkcje, które pobierają dane z programu głównego (te fragmenty danych nazywane są parametrami).