# ENKODER


Enkoder służy do określania:

położenia, kąta, kierunku, prędkości obrotu

## Enkodery przyrostowe

1. Generują impulsy w trakcie obrotu pokrętła
2. Nie podają bezpośrednio aktualnej pozycji
3. Posiadają dwa wyjścia: A i B, które generują sygnały przesunięte w fazie o 90°.

## Enkodery absolutne

1. Podają dokładną pozycję wału w jednym obrocie, co pozwala określić położenie bez konieczności zliczania impulsów.
2. Często są bardziej złożone i droższe od enkoderów przyrostowych.

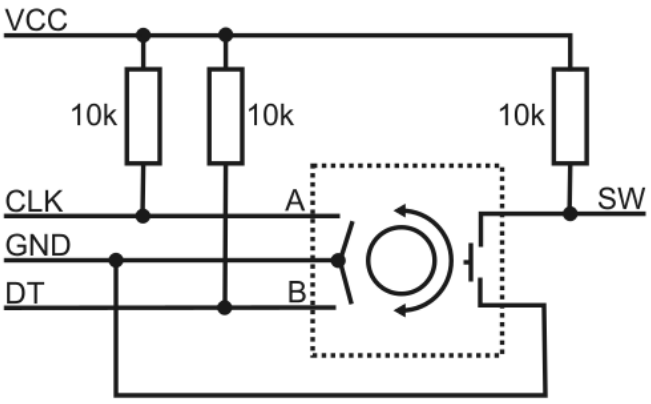
## Rozdzielczość

Rozdzielczość mówi o tym, ile impulsów zostanie wygenerowanych podczas jednego obrotu, tym samym definiuje kąt obrotu.

Enkoder z rozdzielczością 20 imp./obr. wyznacza kąt obrotu równy:

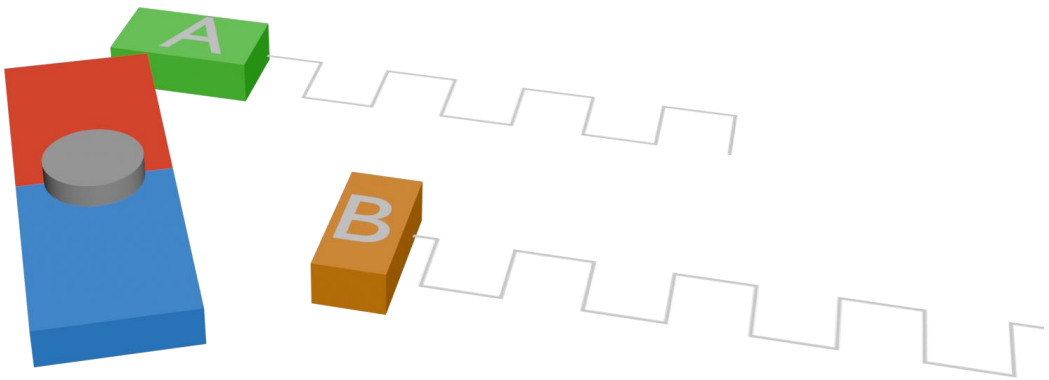
$360^\circ / 20 = 18^\circ$.

# ENKODER
## MAGNETYCZNY



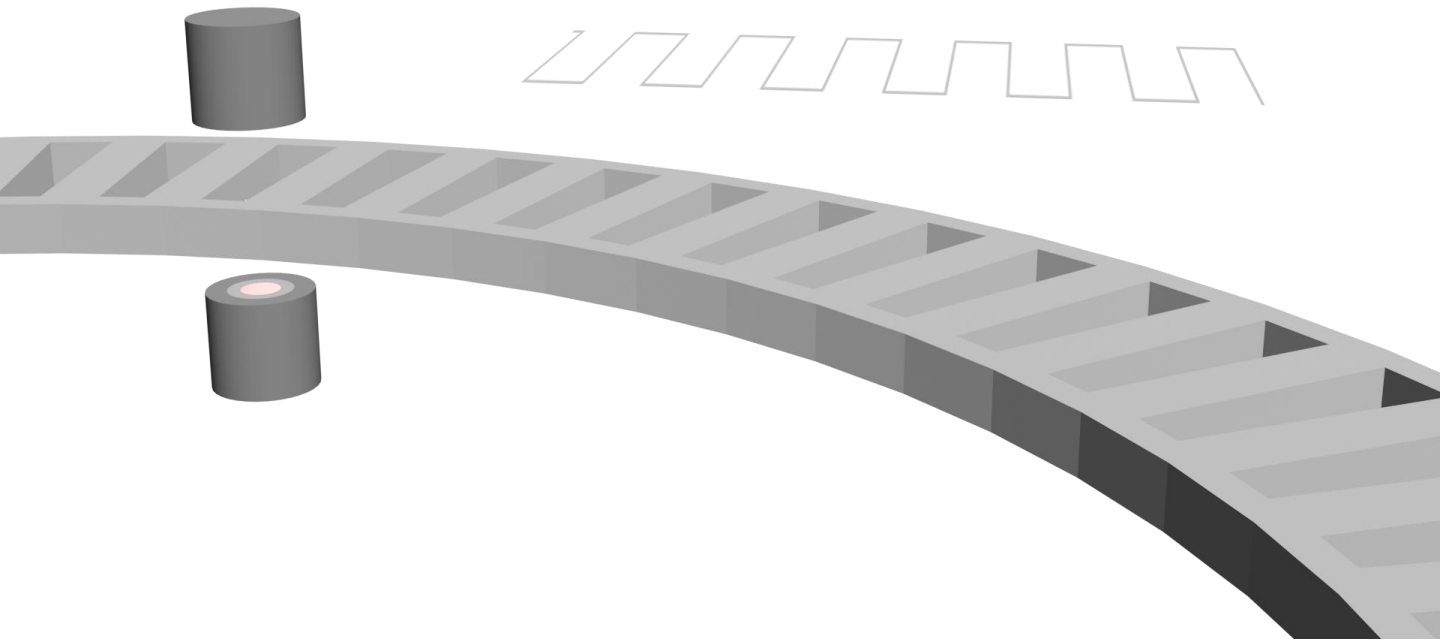

Enkodery stosowane z Arduino wymagają rezystorów podciągających.

### Jak działa przyrostowy enkoder magnetyczny?




Enkodery magnetyczne działają na podstawie efektu Halla. Obracający się magnes wzbudza sygnał na czujniku. Enkodery wyposażone w 2 czujniki (A i B) pozwalają śledzić kierunek obrotu. Drugi czujnik generuje sygnał przesunięty o 90. Podczas obrotu w prawo, sygnał wysoki w pierwszej kolejności powstaje na czujniku A, przy obrocie w lewo jest odwrotnie. Zaczytywanie i porównywanie tych zmian pozwala określić kierunek obrotu.

# ENKODER
## OPTYCZNY



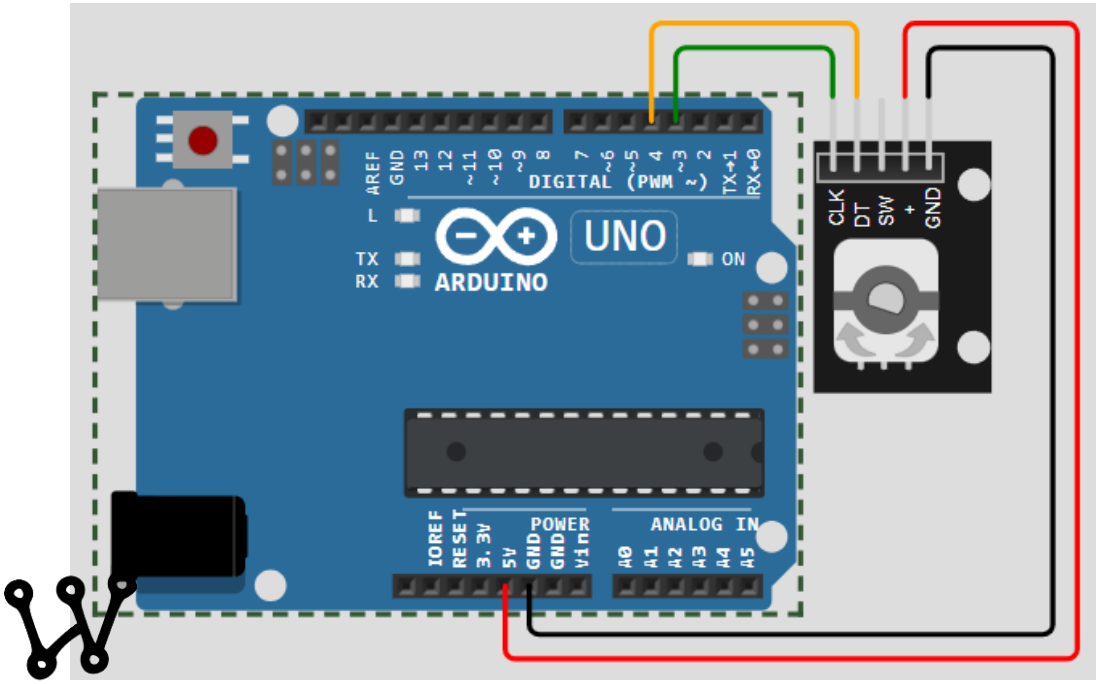
### Jak działa przyrostowy enkoder optyczny?

Enkoder optyczny działa na podstawie strumienia światła (dioda lub laser).

Obracająca się tarcza zawiera miejsca przezroczyste i nieprzezroczyste, które na przemian przepuszczają wiązkę światła.

Poniżej tarczy znajduje się fototranzystor lub fotodioda, która generuje sygnał wysoki po wykryciu wiązki światła.

# ENKODER




Program: Zliczanie impulsów podczas obrotu enkodera w obu kierunkach

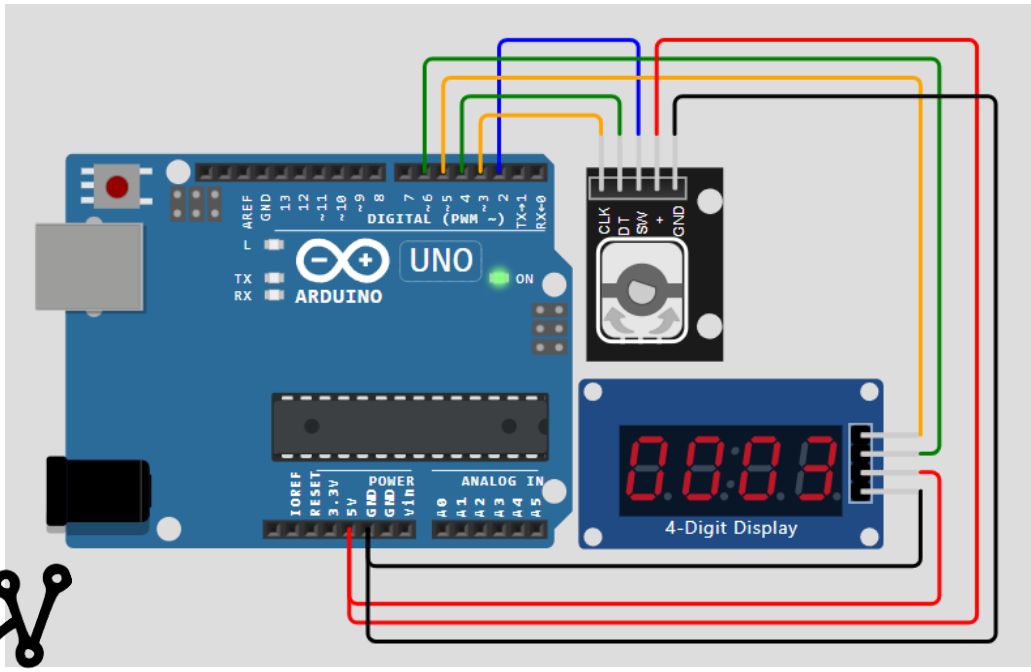
```
#define CLK 3
#define DT 4
int counter = 0;

void setup() {
  pinMode(CLK, INPUT);
  pinMode(DT, INPUT);
  Serial.begin(9600);
}

void loop() {
 readEncoder();
}

void readEncoder(){
 static int lastStateCLK = digitalRead(CLK);
 int currentStateCLK = digitalRead(CLK);
  if (currentStateCLK != lastStateCLK && currentStateCLK == 1) {
    if (digitalRead(DT) != currentStateCLK) {
      counter--;
      if (counter < 0) counter = 0;
    } else {
      counter++;
      if (counter > 20) counter = 20;
    }
    Serial.println(counter);
  }
  lastStateCLK = currentStateCLK;
}
```

ENKODER




Program: Timer z menu ustawiającym czas odliczania wyświetlany na TM1637

```
#include <TM1637Display.h>
#define CLK 3
#define DT 4
#define SW 2
#define CLK_DISPLAY 5
#define DIO 6
TM1637Display display(CLK_DISPLAY, DIO);

class Astratimer {
private:
  int _countDelay;
  unsigned long _previousMillis;
public:
  Astratimer(int countDelay) : _countDelay(countDelay),
_previousMillis(0) {}
  bool canExecute() {
    if (millis() - _previousMillis >= _countDelay) {
      _previousMillis = millis();
      return true;
    }
    return false;
  }
};
Astratimer tim1(1000);
int counter = 0;
int interval = 20;
volatile int menuState = -1;
```

```
void displayNumber(int num) {
  display.showNumberDec(num, true);
}

void toggleMenu() {
  menuState *= -1;
}

void countDown() {
  if (counter > 0) {
    counter--;
    displayNumber(counter);
  } else {
    counter = interval;
    displayNumber(counter);
  }
}

void setup() {
  pinMode(CLK, INPUT);
  pinMode(DT, INPUT);
  pinMode(SW, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(SW), toggleMenu, FALLING);
  display.setBrightness(1);
  display.clear();
}

void loop() {
  static int lastStateCLK = digitalRead(CLK);
  int currentStateCLK = digitalRead(CLK);

  if (menuState == 1) {
    if (currentStateCLK != lastStateCLK && currentStateCLK == 1) {
      if (digitalRead(DT) != currentStateCLK) {
        interval--;
        if (interval < 0) interval = 0;
      } else {
        interval++;
        if (interval > 179) interval = 179;
      }
      displayNumber(interval);
      counter = interval;
    }
    lastStateCLK = currentStateCLK;
  } else {
    if (tim1.canExecute()) {
      countDown();
    }
  }
}
```