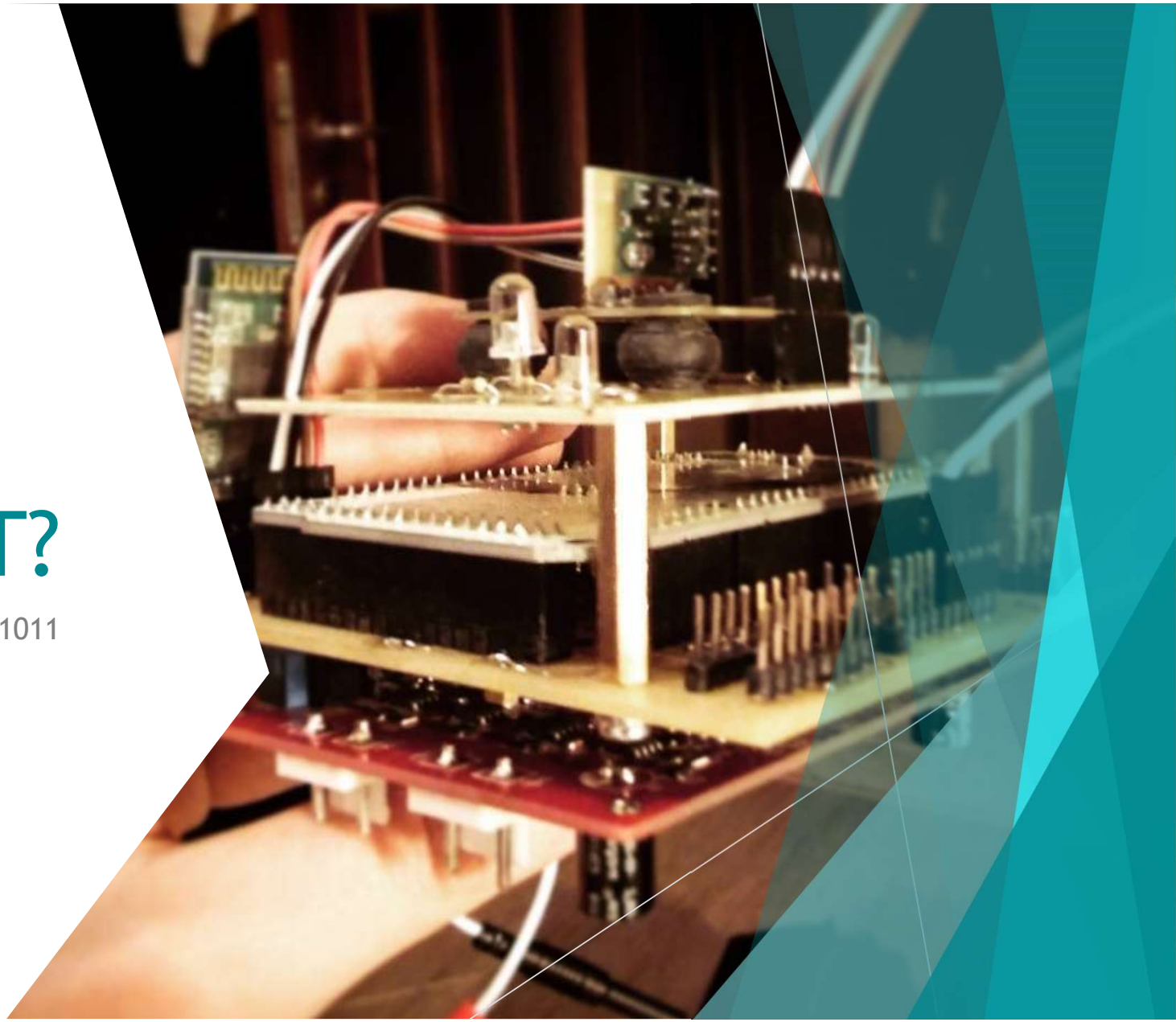
Warsztaty Arduino

Spotkanie #2

@kncybair @CybAiR

Co to UART?

0101101011



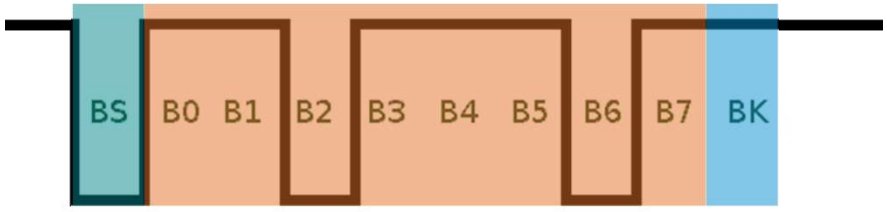
Universal Asynchronous Receiver-Transmitter

UART jest układem scalonym, który zapewnia asynchroniczną komunikację.
W tego typu transmisji jest wymagane by ustalić **szybkość transmisji**, **długość słowa**, **bit parzystości** i **bit stopu**.

9600/8-N-1

Baudrate/Długość słowa-Nieparzystość-Bit Stopu = 1

Poniżej jest przedstawiona ramka **8-bit**owego słowem z bitem startu **BS = 0** i bitem stopu **BK = 1**.



Źródło: forbot.pl

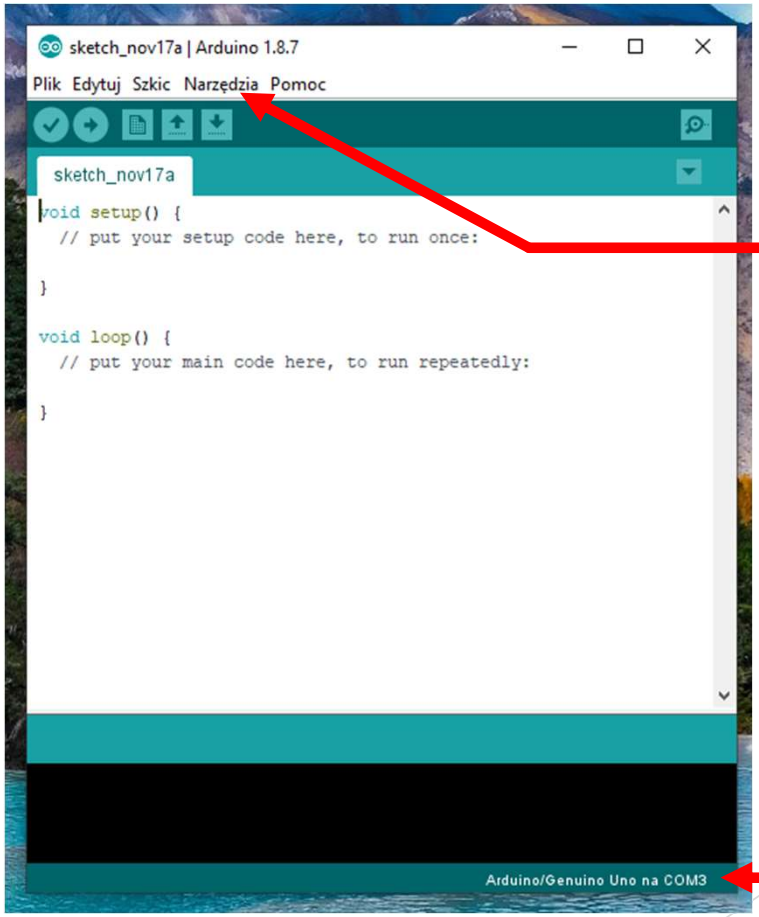
UART vs. USART

UART ma z góry założoną przepustowość o której musi wiedzieć odbiornik, natomiast **USART** jest w stanie sam zsynchronizować nadajnik i odbiornik. **UART** jest jedynie asynchroniczny, a **USART** może zarówno być synchroniczny i asynchroniczny

UART kalkuluje sobie częstotliwość taktowania danych **w oparciu o swój mikrokontroler** i z tym synchronizuje swoje bity startu. **USART** ma natomiast możliwość wyliczenia tego przy Pomocy zewnętrznego zegara i osiągnięcia tym samym **wyższych prędkości przesyłu**.

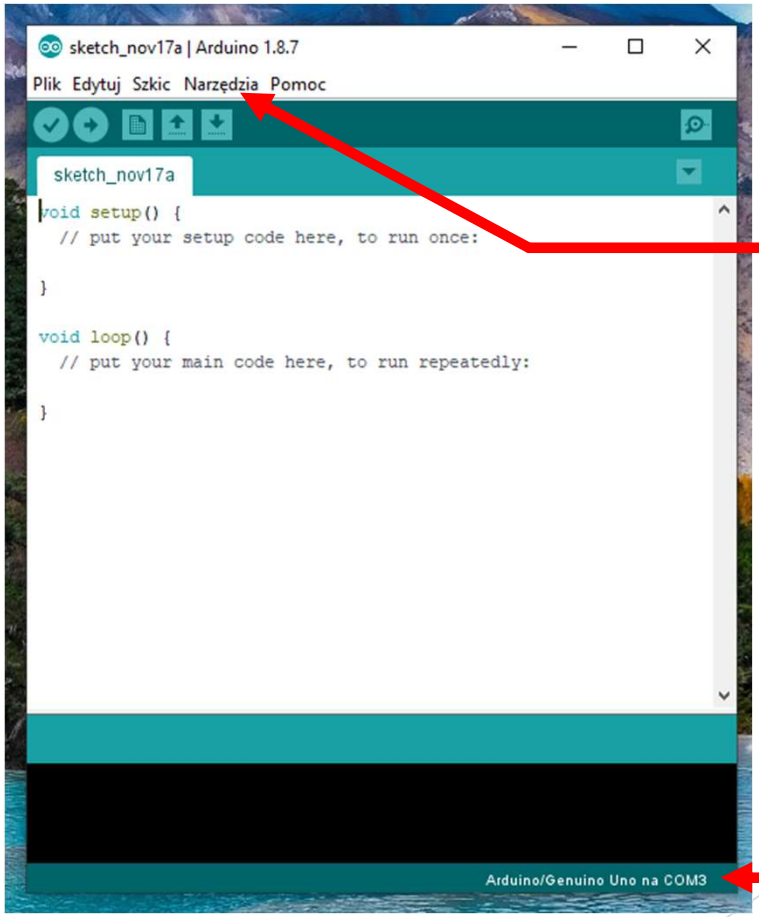
Porty szeregowe w Arduino

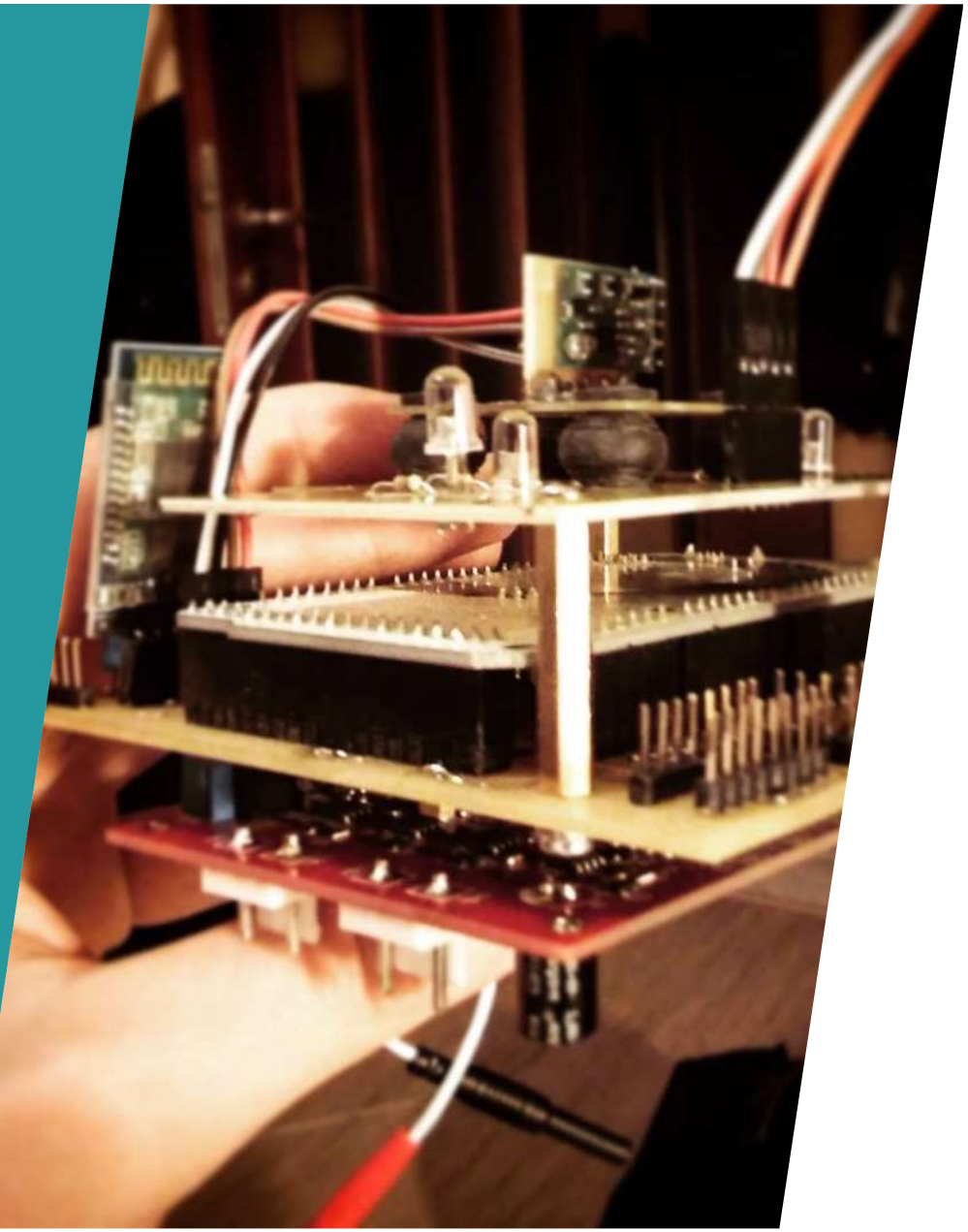
W obecnych czasach komputery są coraz rzadziej wyposażane w oryginalne złącza COM ze względu na szybkość i opłacalność innych rozwiązań jak chociaż USB. Jednak sama nazwa jest wciąż wykorzystywana jako uogólnienie portu szeregowego w niektórych wypadkach.



Porty szeregowe w Arduino

W obecnych czasach komputery są coraz rzadziej wyposażane w oryginalne złącza COM ze względu na szybkość i opłacalność innych rozwiązań jak chociaż USB. Jednak sama nazwa jest wciąż wykorzystywana jako uogólnienie portu szeregowego w niektórych wypadkach.





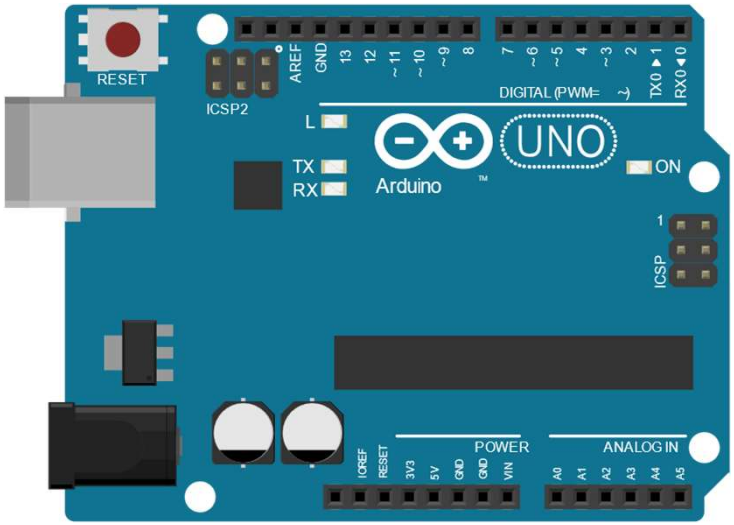
Budujemy!

Projekty z UART

Komunikacja szeregowa

Projekty #1 #2 #3 #4, Schemat
Licznik i przeładowanie zmiennej
Precyzja liczb niecałkowitych
Prezentacja różnych systemów liczbowych

- Potrzebne części:
  - Arduino UNO R3

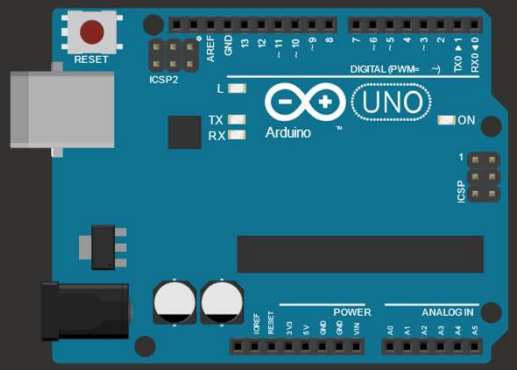


github.com/filesmuggler/octopus

Komunikacja szeregowa

Projekt #1, Kod

Licznik i przeładowanie zmiennej



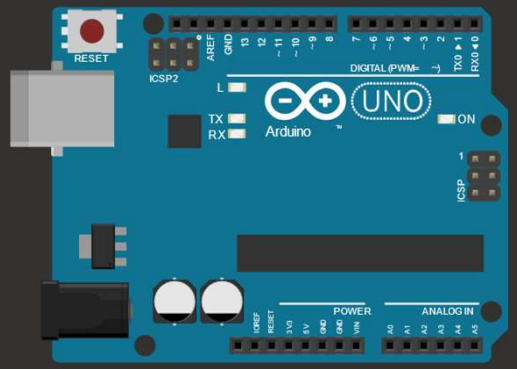

```
#define BAUDRATE 9600
//int counter = 0;
//short counter = 0;
//double counter = 0;
char counter = 0;

void setup() {
    Serial.begin(BAUDRATE);
}

void loop() {
    Serial.print(counter);
    Serial.print(" ");
    counter++;
    delay(5);
}
```

github.com/filesmuggler/octopus

Komunikacja szeregowa

Projekt #2, Kod

Prezentacja różnych systemów liczbowych i alfabetu



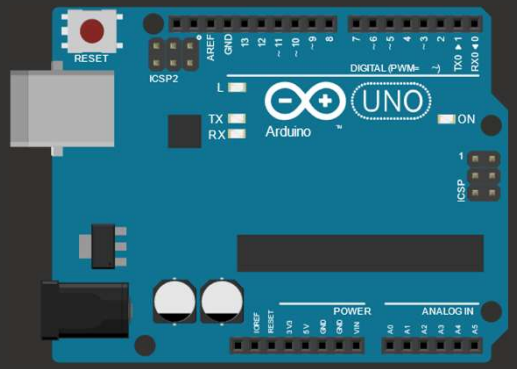
github.com/filesmuggler/octopus

```
#define BAUDRATE 9600

void setup() {
      Serial.begin(BAUDRATE);
      Serial.println("Numbers:");
      for(int i = 0; i < 20; i++){
            Serial.print(i);
            Serial.print(" ");
      }
      Serial.println();
      Serial.println("Alphabet:");
      for(int i = 65; i < 91; i++){
            Serial.write(i);
            Serial.print(" ");
      }
      Serial.println();
      Serial.println();
      Serial.println("Different number systems:");
      Serial.println("BIN: OCT: DEC: HEX:");
      for(int i = 0; i < 16; i++){
            Serial.print(i, BIN);
            Serial.print(" ");
            Serial.print(i, OCT);
            Serial.print(" ");
            Serial.print(i, DEC);
            Serial.print(" ");
            Serial.print(i, HEX);
            Serial.print("\n");
      }
}

void loop() {
}
```

Komunikacja szeregowa

Projekt #3, Kod

Precyzja liczb niecałkowitych



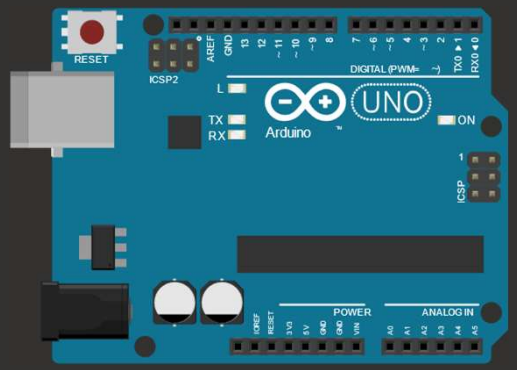
github.com/filesmuggler/octopus

```
#define BAUDRATE 9600
int digits_after_decimal = 3;

float myfloat_1 = 1.2345;
float myfloat_2 = 5.9876;

void setup() {
     Serial.begin(BAUDRATE);

     Serial.print("myfloat1: ");
     Serial.println(myfloat_1, digits_after_decimal);
     Serial.print("myfloat2: ");
     Serial.println(myfloat_2);

     float result = myfloat_1 + myfloat_2;
     Serial.print("result (sum): ");
     Serial.println(result, digits_after_decimal);

     result = myfloat_1 - myfloat_2;
     Serial.print("result (substraction): ");
     Serial.println(result, digits_after_decimal);

     result = myfloat_1 * myfloat_2;
     Serial.print("result (multiplication): ");
     Serial.println(result, digits_after_decimal);

     result = myfloat_1 / myfloat_2;
     Serial.print("result (division): ");
     Serial.println(result, digits_after_decimal);
}

void loop() {}
```

Komunikacja szeregowa

Projekt #4, Kod

Precyzja liczb niecałkowitych




```
#define BAUDRATE 9600
int digits_after_decimal = 10;

void setup() {
    Serial.begin(BAUDRATE);

    Serial.print("PI: ");
    Serial.println(PI);

    Serial.print("PI (extended): ");
    Serial.println(PI, digits_after_decimal);
}

void loop() {
}
```

github.com/filesmuggler/octopus

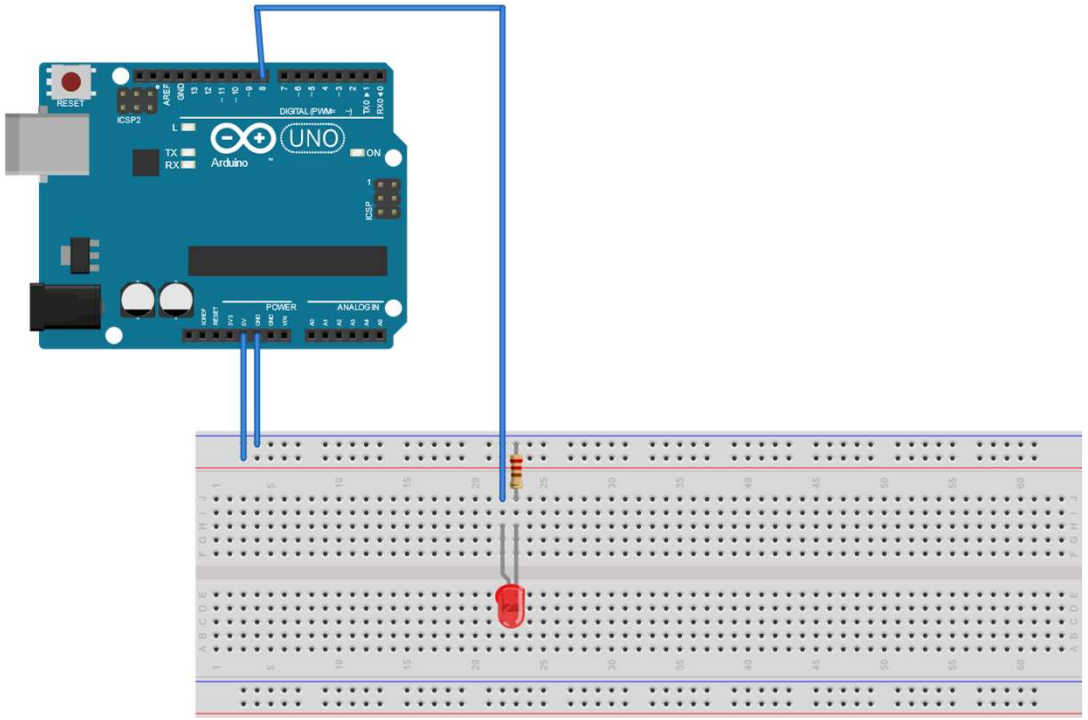
Podpowiedź:

π ≈ 3.141592 653589 793238 462643 383279 502884 197169 399375 ...

Komunikacja szeregowa

Projekty #5 #6, Schemat
Sterowanie diodą przez port szeregowy

- Potrzebne części:
  - Arduino UNO R3
  - Diody LED
  - Rezystory 220 Ohm
  - Przewody

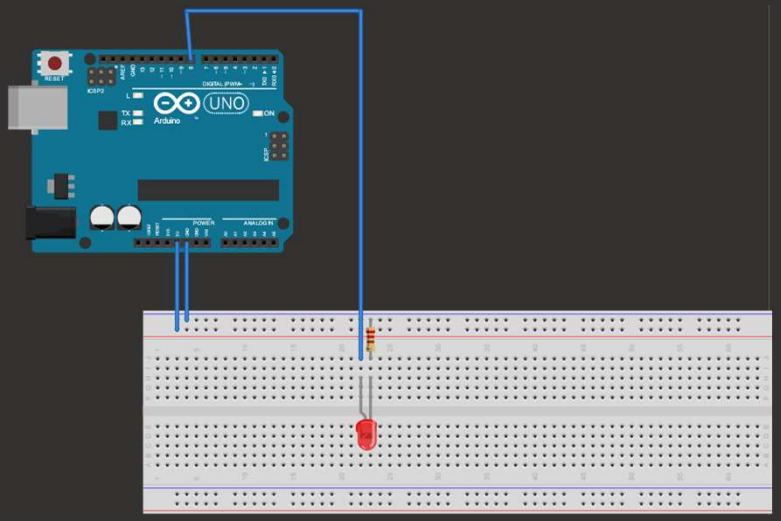


github.com/filesmuggler/octopus

Komunikacja szeregowa

Projekt #5, Kod

Sterowanie diodą przez port szeregowy



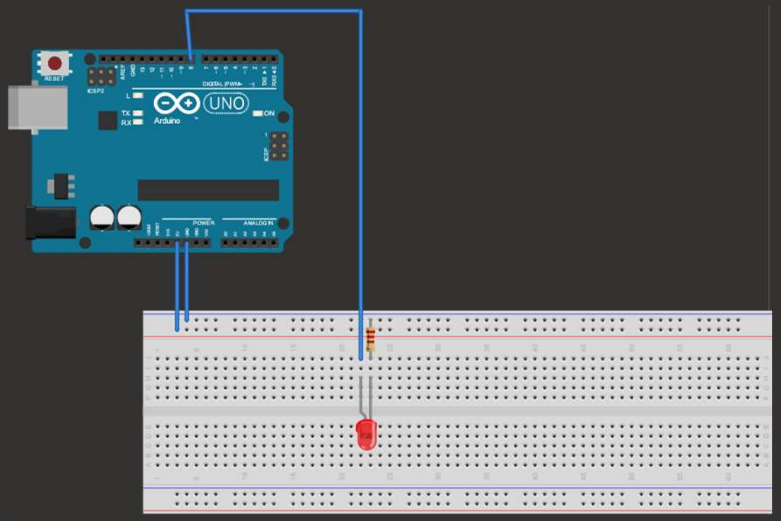
github.com/filesmuggler/octopus

```
#define BAUDRATE 9600
#define LED_PIN 8
String readData;

void setup() {
    Serial.begin(BAUDRATE);
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, LOW);
}

void loop() {
    if(Serial.available()>0){
        readData = Serial.readStringUntil('\n');
        if(readData =="on"){
            Serial.println("Turn on diode");
            digitalWrite(LED_PIN, HIGH);
        }
        else if(readData == "off"){
            Serial.println("Turn off diode");
            digitalWrite(LED_PIN, LOW);
        }
    }
}
```

Komunikacja szeregowa

Projekt #6, Kod – Część pierwsza

Sterowanie diodą przez port szeregowy



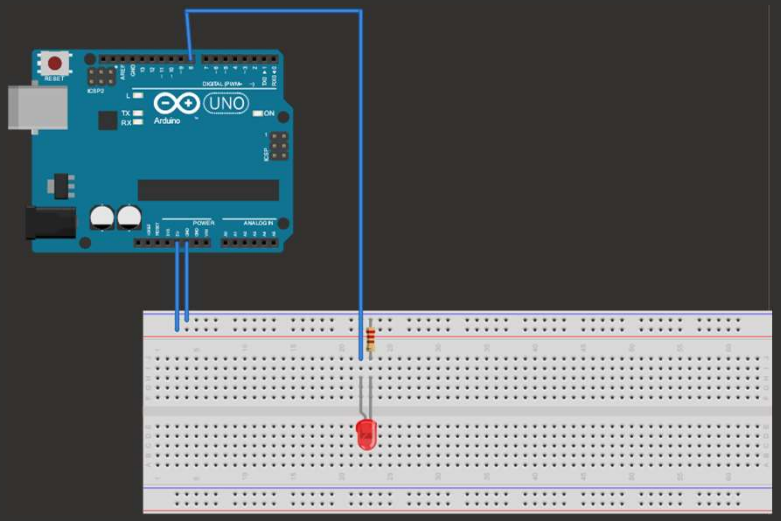
github.com/filesmuggler/octopus

```
#define BAUDRATE 9600
#define LED_PIN 8

String readData;

String userCommand;
String commandValue;

int ledState = LOW;

long previousMillis = 0;

bool if_flicker = false;

long interval = 1000;

void setup() {
    Serial.begin(BAUDRATE);
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, LOW);
}
```

Komunikacja szeregowa

Projekt #6, Kod – Część druga

Sterowanie diodą przez port szeregowy

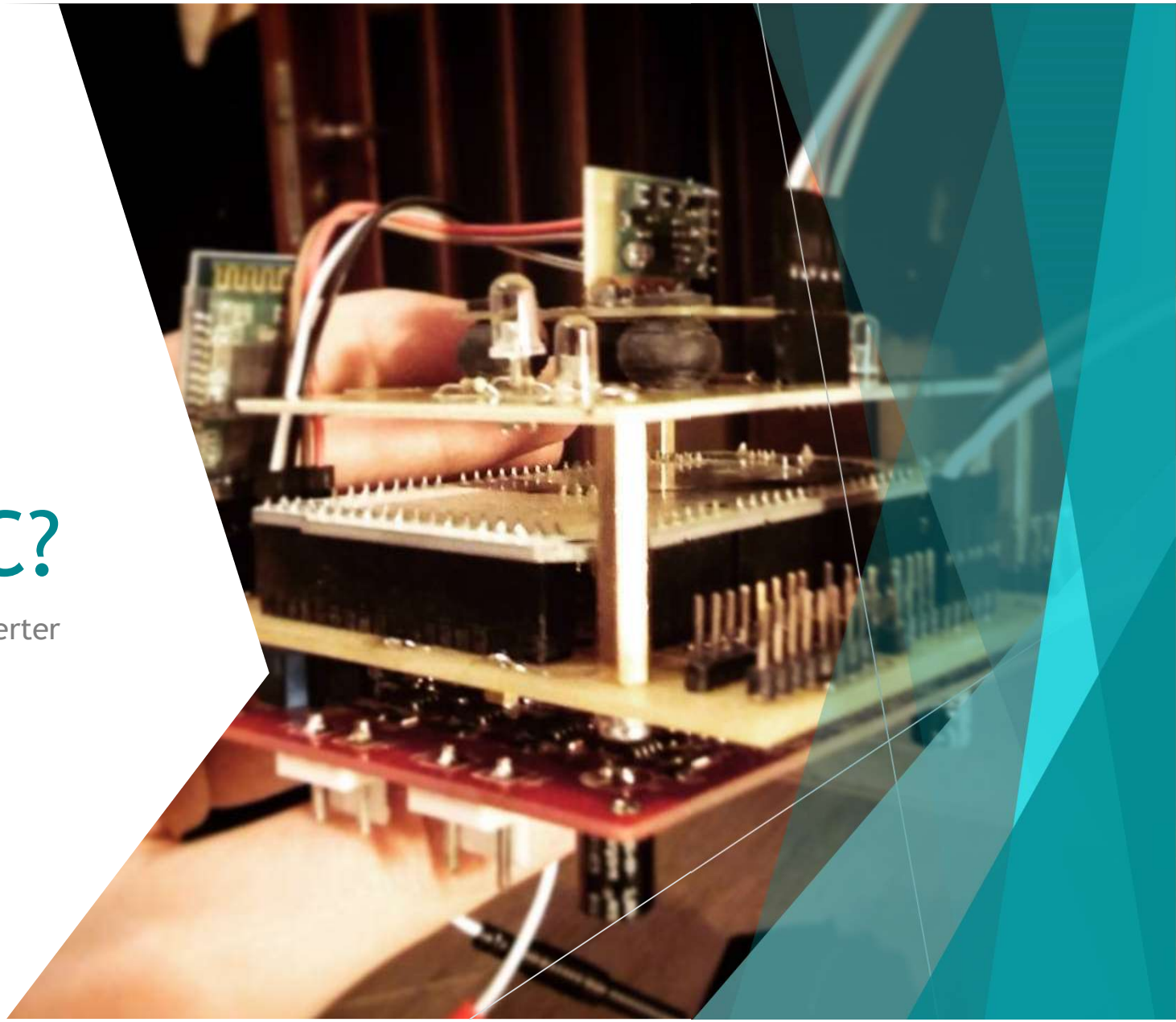


github.com/filesmuggler/octopus

```
void loop() {
      unsigned long currentMillis = millis();
      if(currentMillis - previousMillis > interval && if_flicker) {
            previousMillis = currentMillis;
            if (ledState == LOW){
                  ledState = HIGH;
            }
            else{
                  ledState = LOW;
            }
            digitalWrite(LED_PIN, ledState);
      }

      if(Serial.available()>0){
            readData = Serial.readStringUntil('\n’);
            char sz[] = "Command Value";
            char buf[sizeof(sz)];
            readData.toCharArray(buf, sizeof(buf));
            char *p = buf;
            char *str;
            for(int i = 0; i<2;i++){
                  str = strtok_r(p, " ", &p);
                  if(i==0){
                        userCommand= str;
                  }
                  if(i==1){
                        commandValue = str;
                  }
            }
            if(userCommand == "flicker"){
                  Serial.print("Flickering diode with ms interval: ");
                  Serial.println(commandValue);
                  interval = commandValue.toInt();
                  if_flicker = true;
            }
      }
}
```

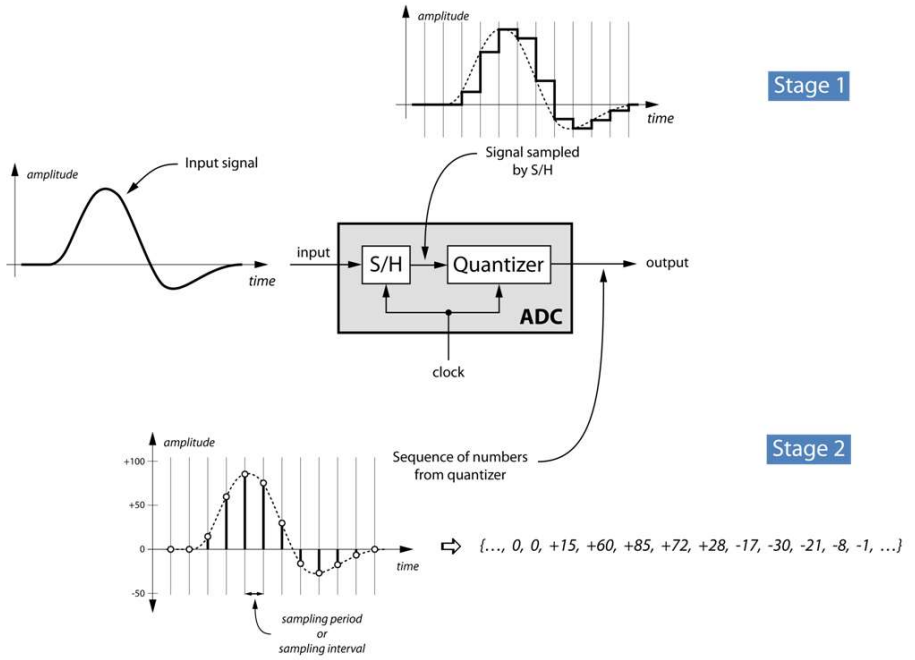
Co to ADC?

Analog-Digital Converter



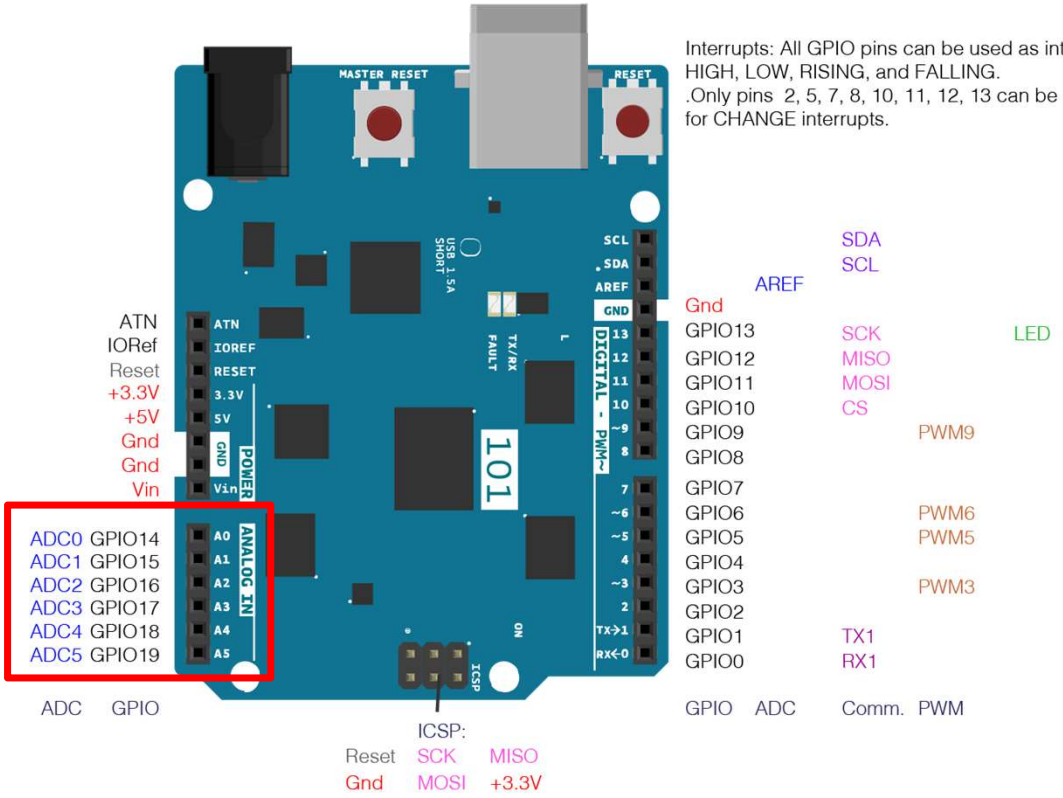
Analog-Digital Converter

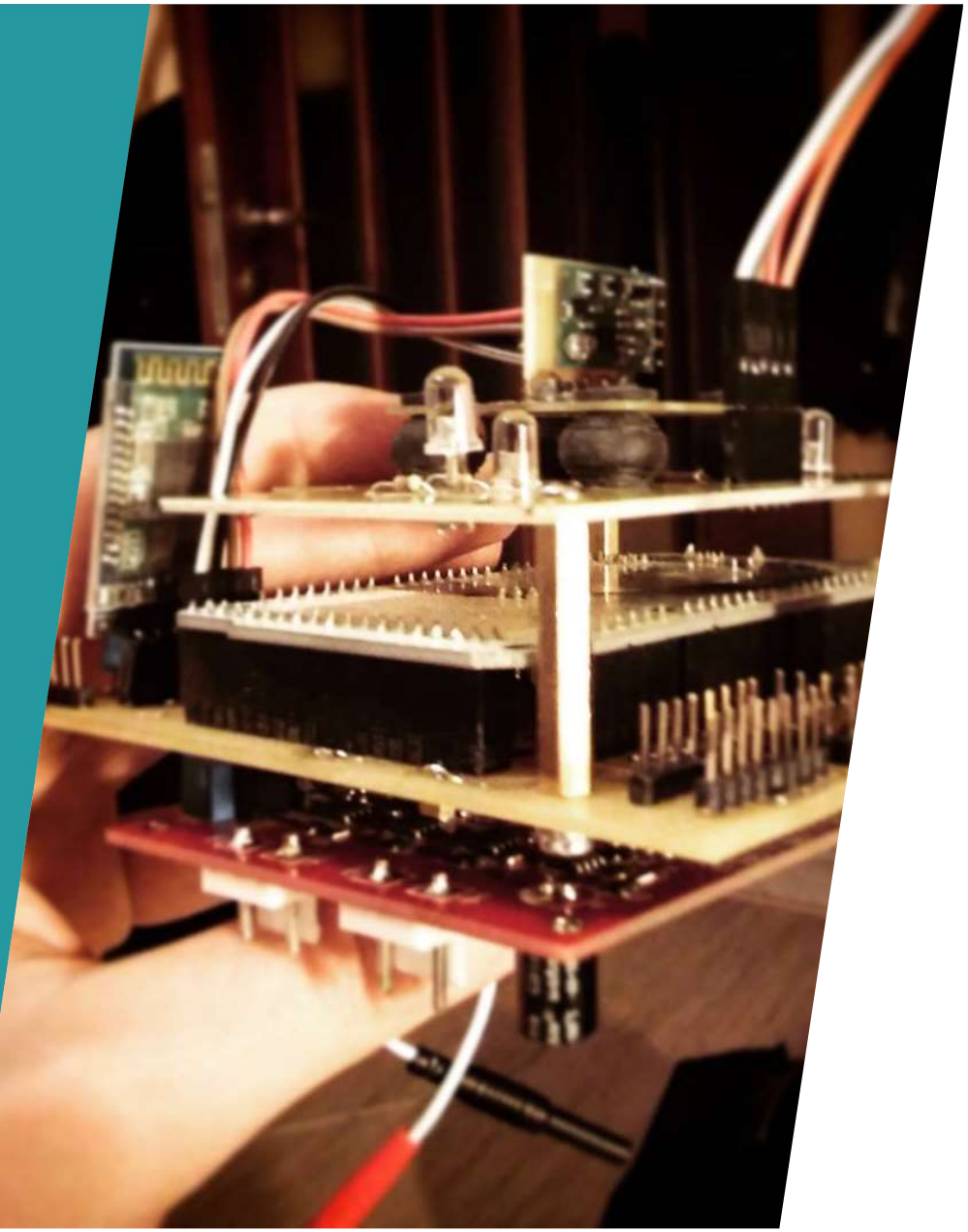
1. Próbkowanie
2. Kwantyzacja
3. Kodowanie



Źródło: nutaq.com

ADC w Arduino





Budujemy!

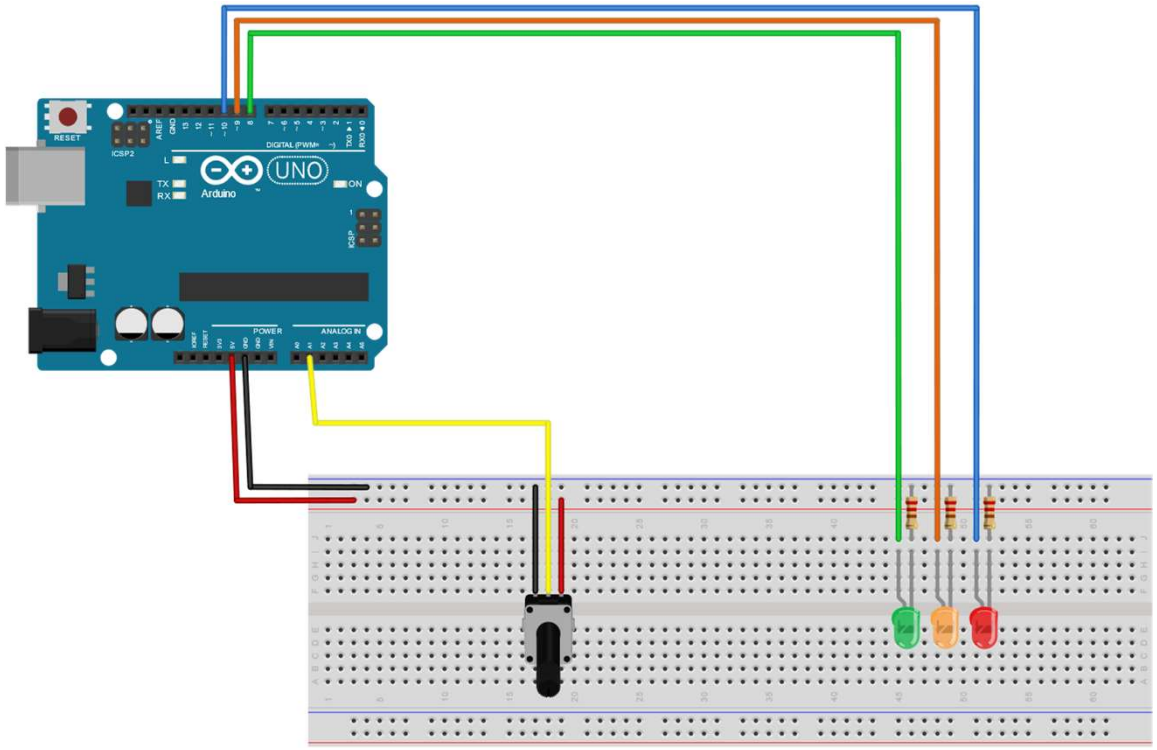
Projekty z ADC

ADC z potencjometrem

Projekty #7 #8, Schemat

Konwersja danych analogowych na cyfrowe

- Potrzebne części:
  - Arduino UNO R3
  - Potencjometr
  - Przewody
  - Diody LED
  - Rezystory 220 Ohm

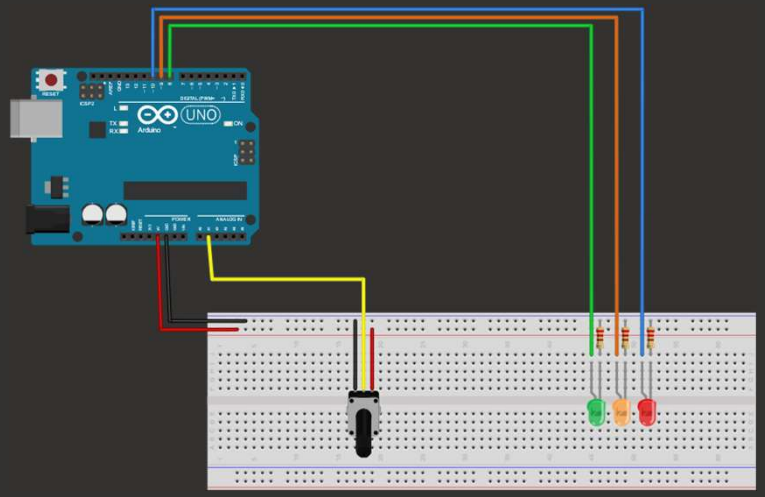


github.com/filesmuggler/octopus

ADC z potencjometrem

Projekt #7, Kod

Konwersja danych analogowych na cyfrowe



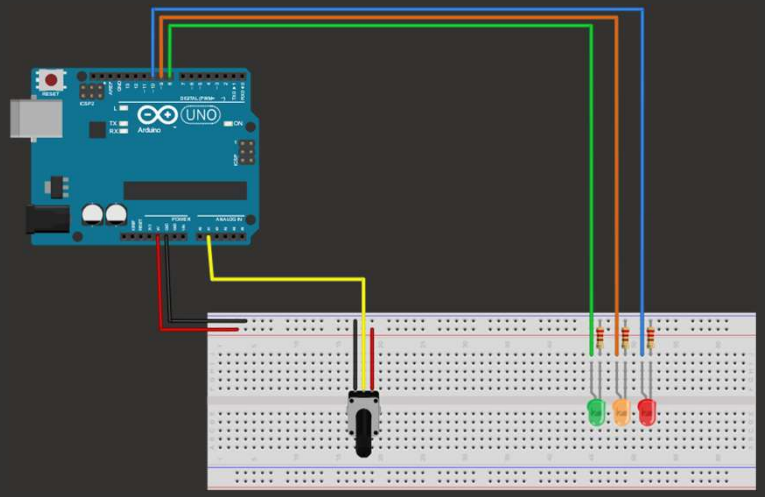
github.com/filesmuggler/octopus

```
#define BAUDRATE 9600
#define GREEN_LED 8
#define YELLOW_LED 9
#define RED_LED 10
#define POTENTIOMETER A1

int pot_value;

void setup(){
      Serial.begin(BAUDRATE);
      pinMode(GREEN_LED, OUTPUT);
      pinMode(YELLOW_LED, OUTPUT);
      pinMode(RED_LED, OUTPUT);
      pinMode(POTENTIOMETER, INPUT);
}

void loop(){
      pot_value = analogRead(POTENTIOMETER);
      Serial.println(pot_value);
      if(pot_value == 0){
            digitalWrite(GREEN_LED,LOW);
            digitalWrite(YELLOW_LED,LOW);
            digitalWrite(RED_LED,LOW);
      }
      else if(pot_value < 350 && pot_value > 0){
            digitalWrite(GREEN_LED,HIGH);
            digitalWrite(YELLOW_LED,LOW);
            digitalWrite(RED_LED,LOW);
      }
      else if(pot_value >350 && pot_value < 700){
            digitalWrite(GREEN_LED,HIGH);
            digitalWrite(YELLOW_LED,HIGH);
            digitalWrite(RED_LED,LOW);
      }
      else {
            digitalWrite(GREEN_LED,HIGH);
            digitalWrite(YELLOW_LED,HIGH);
            digitalWrite(RED_LED,HIGH);
      }
}
```

ADC z potencjometrem

Projekt #8, Kod – Część pierwsza

Konwersja danych analogowych na cyfrowe



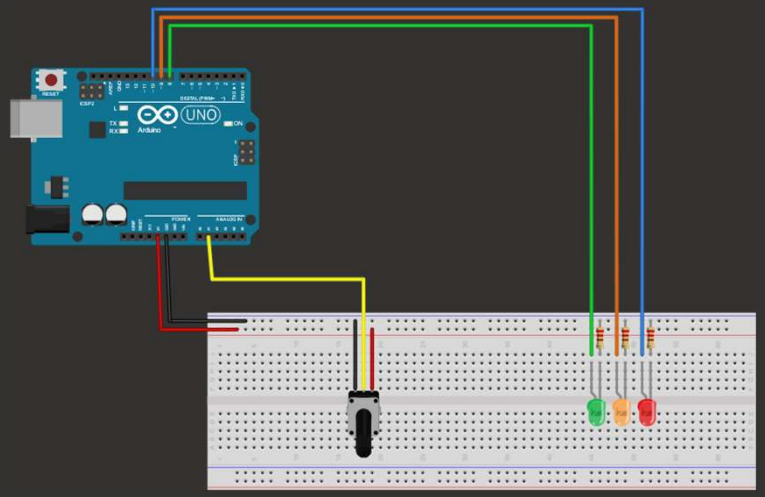
github.com/filesmuggler/octopus

```
#define BAUDRATE 9600
#define GREEN_LED 8
#define YELLOW_LED 9
#define RED_LED 10
#define POTENTIOMETER A1

int pot_value;
int ledState = LOW;
long previousMillis = 0;
long interval = 100;
bool if_blink = false;

void setup(){
    Serial.begin(BAUDRATE);
    pinMode(GREEN_LED, OUTPUT);
    pinMode(YELLOW_LED, OUTPUT);
    pinMode(RED_LED, OUTPUT);
    pinMode(POTENTIOMETER, INPUT);
}
```

ADC z potencjometrem

Projekt #8, Kod – Część druga

Konwersja danych analogowych na cyfrowe



github.com/filesmuggler/octopus

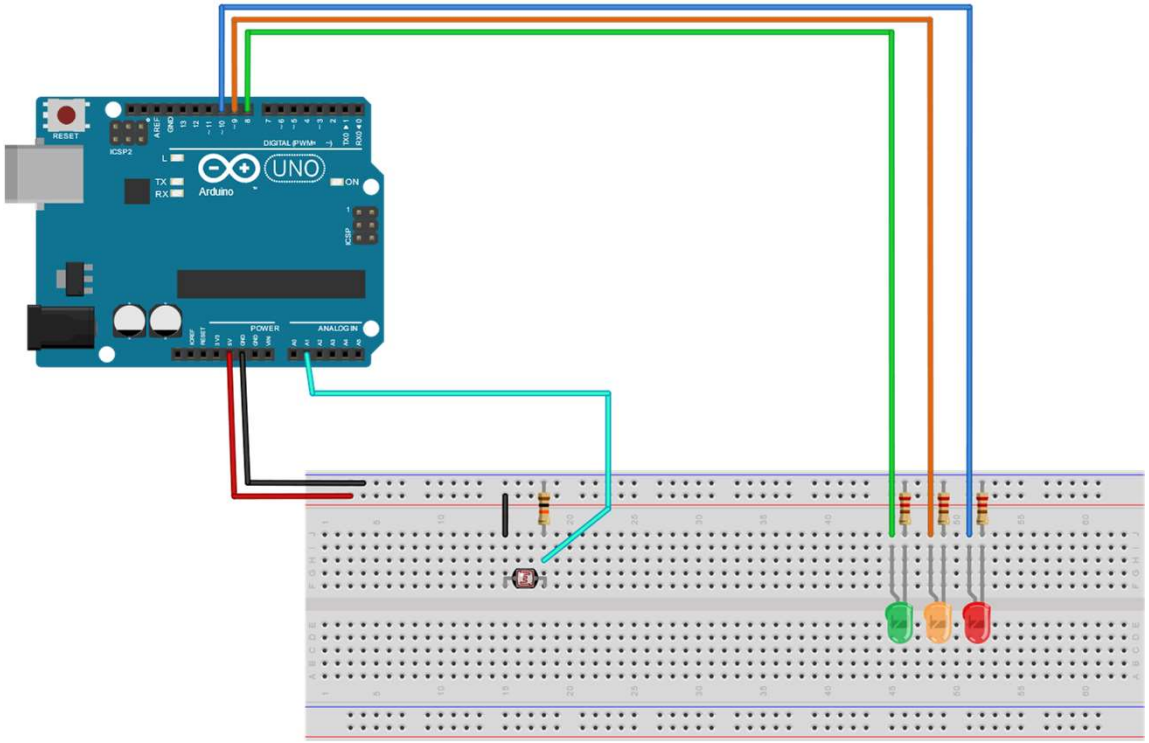
```
void loop(){
	unsigned long currentMillis = millis();
	if(currentMillis - previousMillis > interval && if_blink) {
		previousMillis = currentMillis;
		if (ledState == LOW){
			ledState = HIGH;
		}
		else{
			ledState = LOW;
		}
		digitalWrite(RED_LED, ledState);
	}
	pot_value = analogRead(POTENTIOMETER);
	Serial.println(pot_value);
	if(pot_value == 0){
		if_blink = false;
		digitalWrite(GREEN_LED,LOW);
		digitalWrite(YELLOW_LED,LOW);
		digitalWrite(RED_LED,LOW);
	}
	else if(pot_value < 350 && pot_value > 0){
		if_blink = false;
		digitalWrite(GREEN_LED,HIGH);
		digitalWrite(YELLOW_LED,LOW);
		digitalWrite(RED_LED,LOW);
	}
	else if(pot_value >350 && pot_value < 700){
		if_blink = false;
		digitalWrite(GREEN_LED,HIGH);
		digitalWrite(YELLOW_LED,HIGH);
		digitalWrite(RED_LED,LOW);
	}
	else if(pot_value >700 && pot_value < 1020){
		if_blink = false;
		digitalWrite(GREEN_LED,HIGH);
		digitalWrite(YELLOW_LED,HIGH);
		digitalWrite(RED_LED,HIGH);
	}
	else{
		digitalWrite(GREEN_LED,LOW);
		digitalWrite(YELLOW_LED,LOW);
		if_blink = true;
	}
}
```

ADC z czujnikiem światła

Projekt #9, Schemat

Konwersja danych analogowych na cyfrowe

- Potrzebne części:
  - Arduino UNO R3
  - Fotokomórka
  - Przewody
  - Diody LED
  - Rezystory 220 Ohm i 10 kOhm

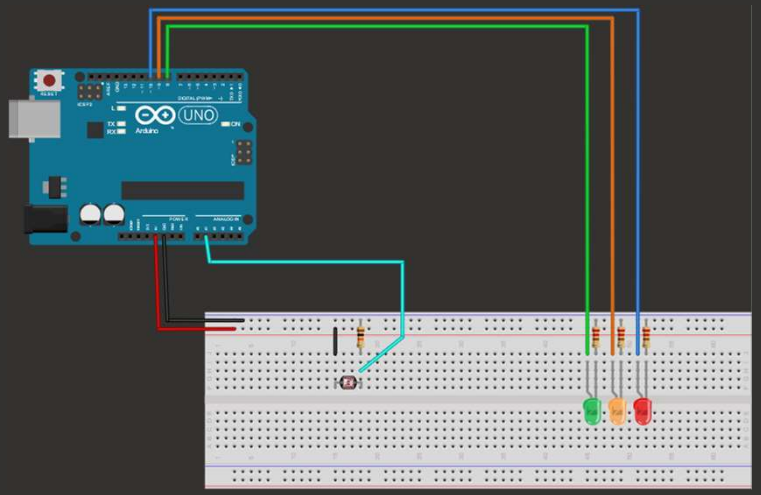


github.com/filesmuggler/octopus

ADC z czujnikiem światła

Projekt #9, Kod

Konwersja danych analogowych na cyfrowe

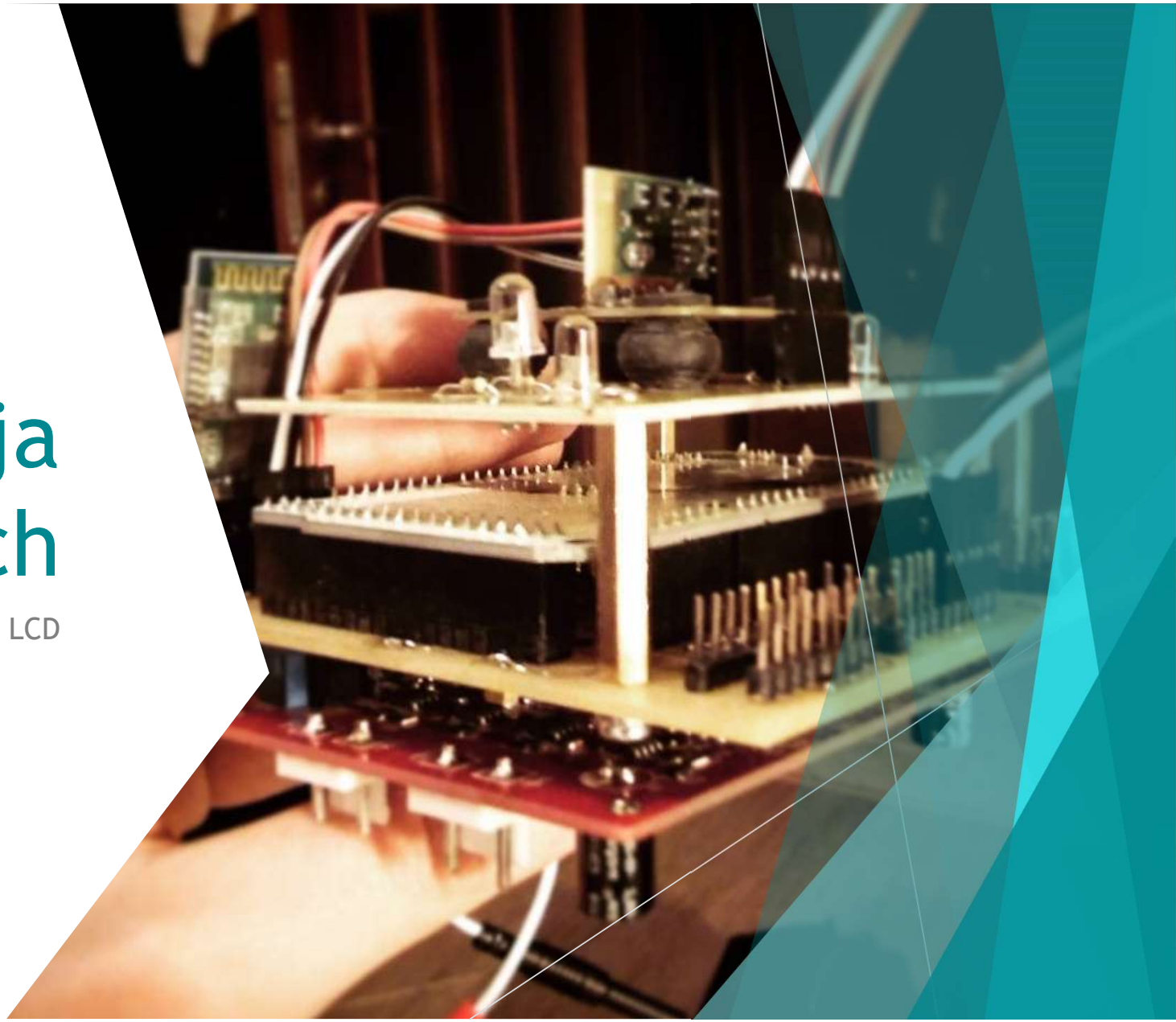


```
#define BAUDRATE 9600
#define SENSOR A0

int sens_value;
int ledPins[] = { 8, 9, 10 };
int ledCount = 3;

void setup(){
      Serial.begin(BAUDRATE);
      for (int thisLed = 0; thisLed < ledCount; thisLed++) {
            pinMode(ledPins[thisLed], OUTPUT);
            digitalWrite(ledPins[thisLed], LOW);
      }
      pinMode(SENSOR, INPUT);
}

void loop(){
      sens_value = analogRead(SENSOR);
      int ledLevel = map(sens_value, 0, 1023, 0, ledCount);

      for (int thisLed = 0; thisLed < ledCount; thisLed++) {
            if (thisLed < ledLevel) {
                  digitalWrite(ledPins[thisLed], HIGH);
            }
            else {
                  digitalWrite(ledPins[thisLed], LOW);
            }
      }
      delay(1000);
}
```

github.com/filesmuggler/octopus

Prezentacja danych

Wyświetlacz LCD



Warsztaty Arduino

github.com/filesmuggler/octopus

@kncybair

@CybAiR