POLITECHNIKA POZNAŃSKA

Wydział Informatyki i Telekomunikacji

# Laboratorium 3.

# Wykorzystanie serwomechanizmów oraz czujniku ruchu i odległości.

Poznań, 2020

## Prawa autorskie



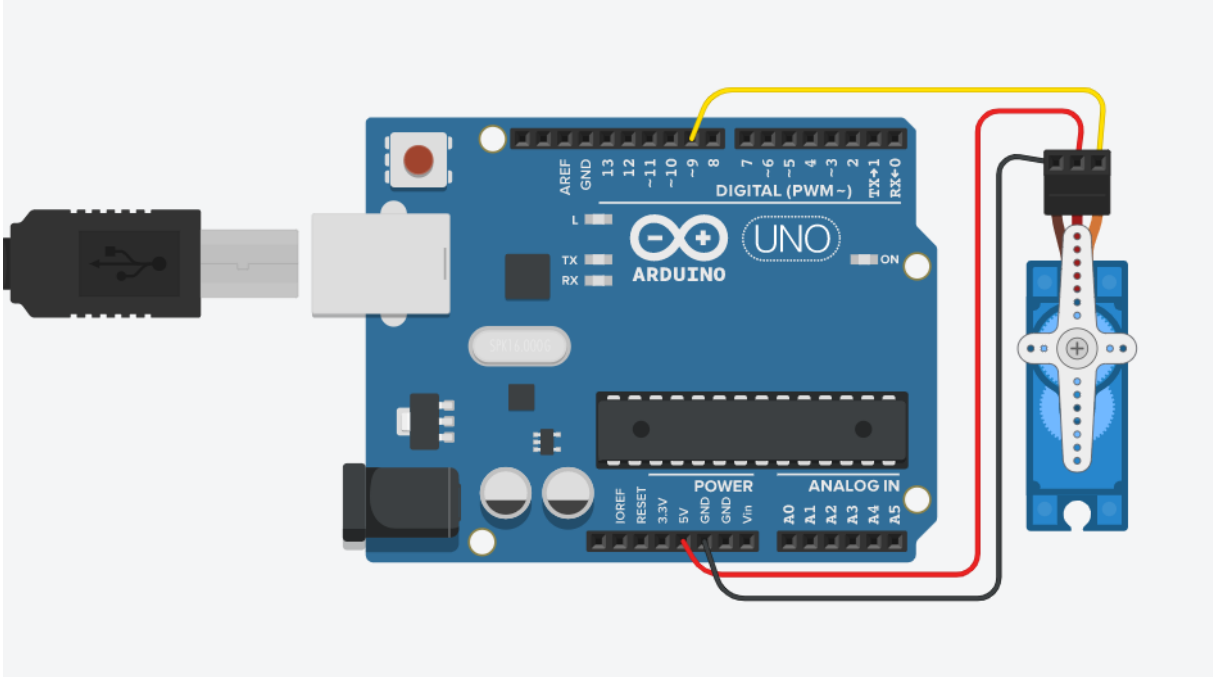
## Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z kolejnymi komponentami, które można wykorzystywać w połączeniu z mikrokontrolerem Arduino oraz przedstawienie ich przykładowych zastosowań w praktyce.

## Przebieg ćwiczenia

W tym ćwiczeniu wykorzystamy głównie serwomechanizmy oraz dwa czujniki – czujnik ruchu i czujnik odległościowy.

## Serwomechanizmy




Prosty program realizujący działanie jednego serwomechanizmu.

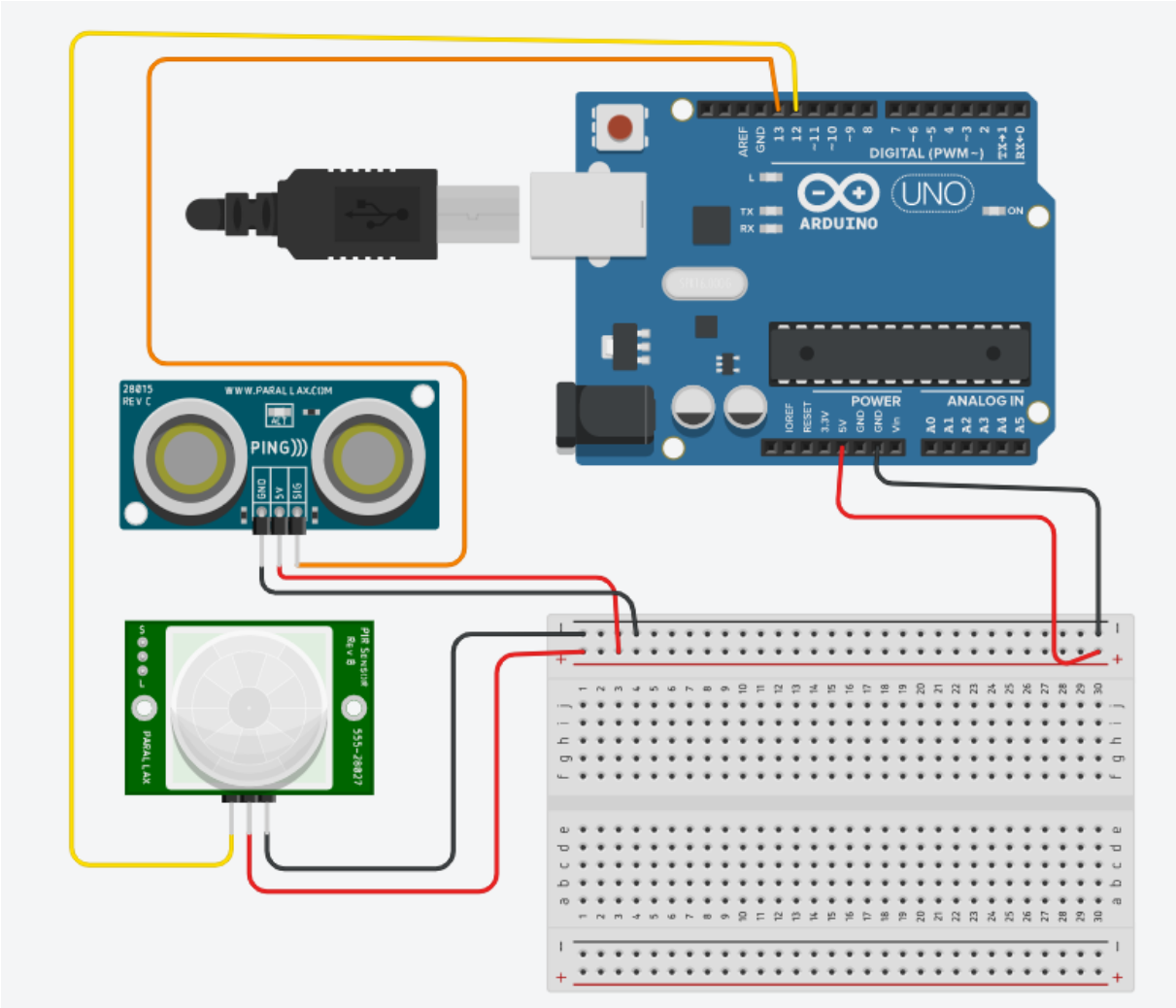
```
#include <Servo.h>
int pos = 0;
Servo servo;


void setup()
{
  servo.attach(9);
}


void loop()
{
  for (pos = 0; pos <= 180; pos += 1) {
     servo.write(pos);
     delay(15);
     }
  for (pos = 180; pos >= 0; pos -= 1) {
       servo.write(pos);
      delay(15);
     }

}
```

## Czujnik ruchu i czujnik odległości




Kod działania czujnika ruchu:

```
#define pirPin 12


int calibrationTime = 30;
long unsigned int lowIn;
long unsigned int pause = 5000;
boolean lockLow = true;
boolean takeLowTime;
int PIRValue = 0;


void setup()
{
    Serial.begin(9600);
    pinMode(pirPin, INPUT);
    }
```

```
void loop()
{
PIRSensor();
}


void PIRSensor()
{
    if(digitalRead(pirPin) == HIGH)
    {
          if(lockLow)
          {
           PIRValue = 1;
           lockLow = false;
           Serial.println("Motion detected.");
           delay(50);
          }
          takeLowTime = true;
    }
   if(digitalRead(pirPin) == LOW)
    {
          if(takeLowTime){lowIn = millis();takeLowTime = false;}
          if(!lockLow && millis() - lowIn > pause)
          {
                PIRValue = 0;
                lockLow = true;
                Serial.println("Motion ended.");
                delay(50);
          }
    }
}
```

Kod działania czujnika odległości:

```
const int odlPin = 13;

void setup() {

     Serial.begin(9600);

      }


void loop()
{
  long duration, cm;
  pinMode(odlPin, OUTPUT);
  digitalWrite(odlPin, LOW);
  delayMicroseconds(2);
  digitalWrite(odlPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(odlPin, LOW);
```

```
  pinMode(odlPin, INPUT);
  duration = pulseIn(odlPin, HIGH);
  cm = microsecondsToCentimeters(duration);


  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(100);
}
    long microsecondsToInches(long microseconds)
{
    return microseconds / 74 / 2;
}
    long microsecondsToCentimeters(long microseconds)
{
    return microseconds / 29 / 2;
}
```

## Zadania do wykonania

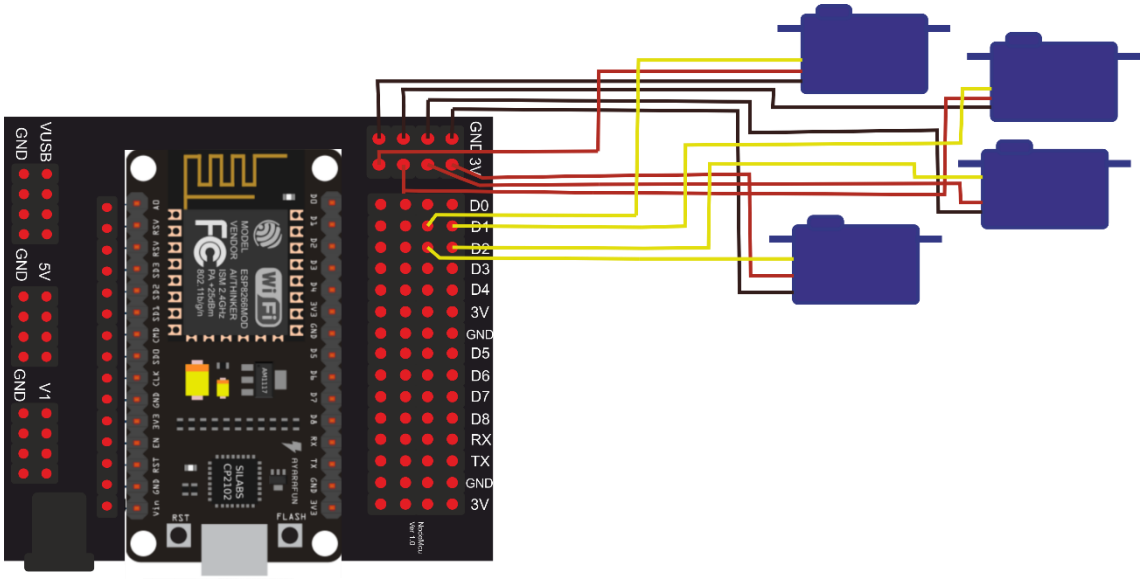
### Zad.1. Brama

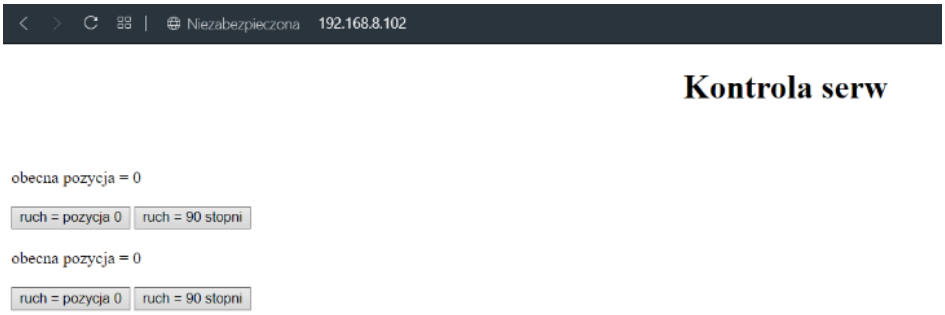
Wykorzystując czujnik ruchu, serwomechanizm oraz diod, zaprojektuj i zaimplementuj prosty symulator bramy wjazdowej. W momencie kiedy czujnik ruchu wykryje ruch, brama (serwomechanizm) powinna zacząć się otwierać oraz migać zielona dioda. Po upływie 15sek. Braku wykrycia ruchu, brama ma się zamykać i migać na czerwono. Opcjonalnie można dodać do tego sygnał dźwiękowy.

### Zad. 2.

Zaproponuj i opracuj własny projekt z wykorzystaniem komponentów, które są dostępne w symulatorze, a nie zostały przedstawione w zadaniach laboratoryjnych.

## Dodatek




Platformę NodeMCU V3 podłączono do dedykowanej płytki testowej. Rozszerza ona wyprowadzenia GPIO, oraz umożliwia zasilanie urządzeń peryferyjnych napięciem 3V i 5V w zależności od zapotrzebowania. Sama płytka zasilana może być prądem stałym od 6-24V. Cztery serwa zostały podłączone pod dwa piny. Serwa te zostały wykorzystane do wprawienia w ruch model 3D, stąd takie połączenie. Dwa serwa odpowiadające za ruch modelu na boki podłączono do pinu D1(GPIO 5), pozostałe dwa serwa, które poruszają modelem w górę/dół podłączono do pinu D2(GPIO 4). Sposób połączenia przedstawiono na powyższym schemacie. Zasilane są napięciem 3V z płytki testowej.

Aby zwiększyć kontrolę nad poruszaniem się modelu, dodano do programu możliwość sterowania serwami za pomocą prostej strony. Strona może być uruchomiona z każdego urządzenia które znajduję się w tej samej sieci co płytka. Na stronie przedstawiono aktualną pozycję serwomechanizmów oraz zaimplementowano przyciski pozwalające zmieniać ich

pozycję. Pozycję można zmieniać z pozycji zerowej o 90º. Na zrzucie nie zostało to ujęte, aczkolwiek sam program posiada jeszcze możliwość włączenia automatycznej symulacji przedstawiające możliwości ruchu serw w modelu.

```
#include <ESP8266WiFi.h>
#include <Servo.h>


Servo servo;
Servo servo1;


int licznik=1;


const char* ssid = "PLAY INTERNET 4G LTE-F928";
const char* password = "93642682";


WiFiServer server(80);


void setup() {
  Serial.begin(115200);
  delay(10);
  servo.attach(5); //port D1
  servo1.attach(4); //port D2


  Serial.println();
  Serial.println();
  Serial.print("Łączenie z  ");
  Serial.println(ssid);


  WiFi.begin(ssid, password);


  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  Serial.println("");
  Serial.println("połączono");


  // Start serwera
  server.begin();
 Serial.println("Start serwera");
}


void loop() {
  // sprawdzenie czy klient jest połączony z siecią
  WiFiClient client = server.available();
  if (!client) {
    return;
  }


  // czekaj kiedy klient przesyła dane
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  }

  String request = client.readStringUntil('\r');
  Serial.println(request);
  client.flush();


 int wart = 0;
 int wart1 = 90;

 //dopasowanie żądań

//serwa z D2
if (request.indexOf("/poz=0") != -1)  {
  servo.write(0); //ruch serwa na pozycje 0
  wart=0;
}
```

```
if (request.indexOf("/poz=90") != -1)  {
  servo.write(90); //ruch serwa na 90 stopni
  wart=90;
}

//serwa z D3
if (request.indexOf("/pozz=0") != -1)  {
  servo1.write(0); //ruch serwa na pozycje 0
  wart1=0;
}

if (request.indexOf("/pozz=90") != -1)  {
  servo1.write(90); //ruch serwa na 90 stopni
  wart1=90;
}


//////////////////symulacja/////
if (request.indexOf("/sym") != -1){
 for (licznik=1;licznik<6;++licznik)
 {
    for(wart = 0; wart < 90; wart++)
   {
    servo.write(wart);
     delay(20);
     }
     for(wart = 90; wart > 0; wart--)
    {
    servo.write(wart);
    delay(20);
    }}

 ////////////////////////////////

 for (licznik=6;licznik<11;++licznik)
 {
    for(wart1 = 0; wart1 < 90; wart1++)
     {
    servo1.write(wart1);
```

```
    delay(20);
    }
    for(wart1 = 90; wart1 > 0; wart1--)
    {
     servo1.write(wart1);
    delay(20);
    }
 }
 ////////////////////////////////////////
  for (licznik=11;licznik<14;++licznik)
 {
    for(wart = 0; wart < 90; wart++)
    {
    servo.write(wart);
     delay(20);
     }
     for(wart = 90; wart > 0; wart--)
    {
    servo.write(wart);
    }}
 ////////////////////////////////////////////////////////
 for (licznik=14;licznik<18;++licznik)
 {
    for(wart1 = 0; wart1 < 90; wart1++)
     {
    servo1.write(wart1);
    delay(20);
    }
    for(wart1 = 90; wart1 > 0; wart1--)
    {
     servo1.write(wart1);
    delay(20);
    }
 }
}
```

```
  // zwróc odpowiedz

  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("");
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");
  client.println("<h1 align=center>Kontrola serw</h1><br><br>");
  client.print("obecna pozycja  = ");
  client.print(wart);
  client.println("<br><br>");
  client.println("<a href=\"/poz=0\"\"><button>ruch = pozycja 0
</button></a>");
  client.println("<a href=\"/poz=90\"\"><button>ruch = 90
stopni</button></a>");

  client.println("<br><br>");
  client.print("obecna pozycja  = ");
  client.print(wart1);
  client.println("<br><br>");
  client.println("<a href=\"/pozz=0\"\"><button>ruch = pozycja 0
</button></a>");
  client.println("<a href=\"/pozz=90\"\"><button>ruch = 90
stopni</button></a>");
  client.println("<br><br>");
  client.println("<a
href=\"/sym\"\"><button>symulacja</button></a>");

  client.println("</html>");
  delay(1);
  Serial.println("rozłączono");
  Serial.println("");
}
```