# Założenia projektu oraz wykorzystane komponenty

Głównym założeniem naszego projektu było stworzenie bezprzewodowej linii produkcyjnej za pomocą modułu ESP8266.

Ten oto moduł odpowiada za połączenie ze sobą wszystkich pozostałych modułów, w celu przekazania informacji oraz określonych zadań, które mają zostać wykonane. Wykorzystaliśmy go również do stworzenia serwera

sieciowego, za pomocą którego można zdalnie sterować urządzeniem (szybkość

pracy).

Do stworzenia projektu wykorzystaliśmy moduł sterowania silnikami DC 12V L298N, a same silniki pobierają stosunkowo niewielką moc. Posłużył nam on do sterowania silnikami oraz kontroli ich prędkości obrotów i zasilania. Silniki są zasilane za pomocą zasilacza 2A 5V. Silnik DC12V posiada również analogowy przełącznik polaryzacji przez co można ręcznie zmienić kierunek ruchu linii produkcyjnej.

W projekcie został również zastosowana płytka RFID-RC522, która służy do odczytywania kart mających na celu przekierowywanie dostępu do strony, poprzez którą sterujemy silnikami napędzającymi naszą linię produkcyjną.

W projekcie został również stworzony osobny ośrodek który ma za zadanie

wyliczyć prędkość obrotów silnika. Zostały do tego wykorzystane rasberry pi zero 1.3 oraz sensor podczerwieni

Wykorzystane komponenty w projekcie:

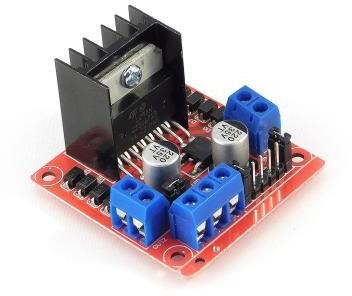
* ESP8266



* RFID-RC522



* L298N



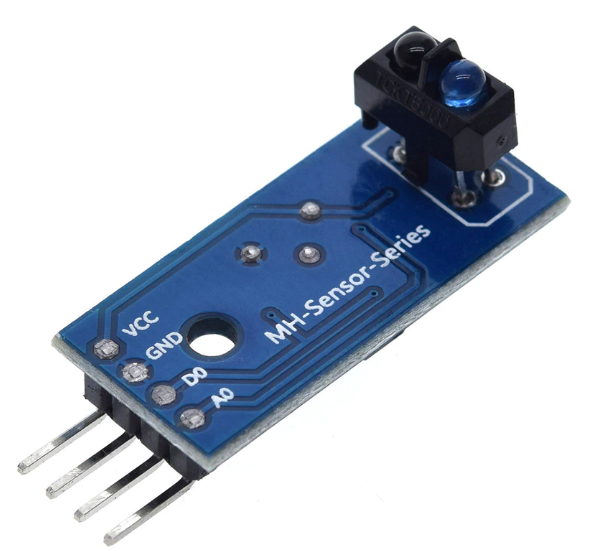
* Zasilacz 5V 2A



* Raspberry Pi zero v1.3



* Sensor podczerwieni



# Działanie oraz opis

Urządzenie zostało umieszczone na stelażu, który utrzymuje naszą linię

produkcyjną, a ta z kolei jest wprawiana w ruch za pomocą silników DC 12V.

Całość została wykonana z części wykorzystanych ze starych zabawek.

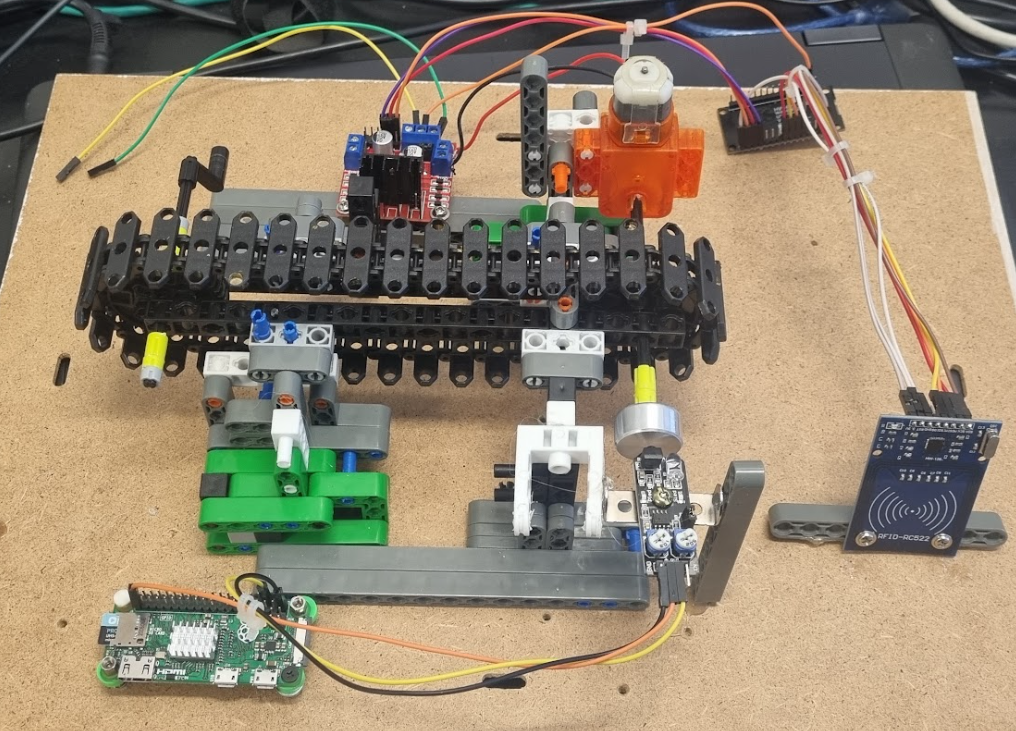
W następnym etapie silnik został podłączony do urządzenia.

Środkowym punktem odniesienia było podłączenie wszystkich potrzebnych

portów L298N do silników (ENA=D1 , IN1=D2 IN2=D2 OUT1=SILNIK

OUT2=SILNIK) zostało podpięte zasilanie opisane powyżej (piny opisane w podpunkcie zostały odpowiednio podłączone do płytki ESP8266 według przepisów powyżej).

Następnym punktem było podpięcie systemu RFID-RC522 (RFID jest zasilany napięciem nie większym niż **3.3V**) do ESP8266 w następującej kolejności (GND- GND 3.3V-3.3V RST-D1 SDA-D8 SCK-D5 MOSI-D6 MISO-D7).



*Zdjęcie 1. Podłączona linia produkcyjna*

[Krótki filmik przedstawiający działanie linii produkcyjnej](https://streamable.com/09ozu5)

# Serwer oraz jego działanie

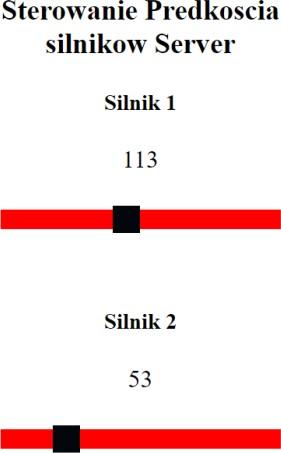
Serwer został postawiony za pomocą płytki ESP8266 oraz bibliotekami

#include <ESP8266WiFi.h> #include <ESPAsyncTCP.h> #include <ESPAsyncWebServer.h>

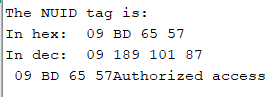
Dzięki temu mogliśmy postawić serwer lokalny na którym można było przedstawić działanie zdalne silników oraz możliwość sterowania nimi na odległość np. za pomocą telefonu komórkowego. Jedynym

wymogiem całej operacji było podłączenie się do sieci, w której został

napisany program. Program wykorzystuje HTML oraz JavaScript



*Zdjęcie 2. Witryna umożliwiająca zdalne sterowanie pracą silników*



*Zdjęcie 3. Punkt dostępu do strony RFID*

# Tworzenie urządzenia pomiarowego

Do stworzenia urządzenia pomiarowego zostało wykorzystane raspberry pi. Do raspberry pi został połączony sensor podczerwieni oraz został stworzony kod który ma za zadanie zliczać dzięki przymocowanej tarczy z czarną kreską stany dzięki czemu można wyliczyć prędkość obrotów silnika

# KOD Programu

//Przypisanie odpowiednich pinów do płytki

int ENA\_pin =D7; int ENB\_pin = D8; int IN1 = D1;

int IN2 = D2; int IN3 = D3; int IN4 = D4;

//Zadeklarowanie zmiennych to odpowiedniego działania silników

String slider\_value = "0"; String slider\_value2 = "0";

const char\* input\_parameter = "value"; const char\* input\_parameter2 = "value2";

//łaczenie z siecią na porcie 80

AsyncWebServer server(80);

////kod HTML , ccs i javascrit który opisuje nam wygląd stroy oraz zasadę działania tej

strony

const char index\_html[] PROGMEM = R"rawliteral(

<!DOCTYPE HTML><html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Sterowanie Predkoscia silnikow Server</title>

<style>

html {font-family: Times New Roman; display: inline-block; text-align: center;} h2 {font-size: 2.3rem;}

h4 {font-size: 1.8rem;} p {font-size: 2.0rem;}

body {max-width: 400px; margin:0px auto; padding-bottom: 25px;}

.slider { -webkit-appearance: none; margin: 14px; width: 360px; height: 25px; background: #FF0000;

outline: none; -webkit-transition: .2s; transition: opacity .2s;}

.slider::-webkit-slider-thumb {-webkit-appearance: none; appearance: none; width: 35px; height: 35px; background:#01070a; cursor: pointer;}

.slider::-moz-range-thumb { width: 35px; height: 35px; background: #01070a; cursor: pointer; }

.slider2 { -webkit-appearance: none; margin: 14px; width: 360px; height: 25px; background: #FF0000;

outline: none; -webkit-transition: .2s; transition: opacity .2s;}

.slider2::-webkit-slider-thumb {-webkit-appearance: none; appearance: none; width: 35px; height: 35px; background:#01070a; cursor: pointer;}

.slider2::-moz-range-thumb { width: 35px; height: 35px; background: #01070a; cursor: pointer; }

</style>

</head>

<body>

<h2>Sterowanie Predkoscia silnikow Server</h2>

<h4>Silnik 1</h4>

<p><span id="textslider\_value">%SLIDERVALUE%</span></p>

<p><input type="range" onchange="updateSliderPWM(this)" id="pwmSlider" min="0" max="255" value="%SLIDERVALUE%" step="1" class="slider"></p>

<br/>

<h4>Silnik 2</h4>

<p><span id="textslider\_value2">%SLIDERVALUE2%</span></p>

<p><input type="range" onchange="updateSliderPWM2(this)" id="pwmSlider2" min="0" max="255" value="%SLIDERVALUE2%" step="1" class="slider2"></p>

<script>

function updateSliderPWM(element) {

var slider\_value = document.getElementById("pwmSlider").value; document.getElementById("textslider\_value").innerHTML = slider\_value; console.log(slider\_value);

var xhr = new XMLHttpRequest();

xhr.open("GET", "/slider?value="+slider\_value, true); xhr.send();

}

function updateSliderPWM2(element) {

var slider\_value2 = document.getElementById("pwmSlider2").value; document.getElementById("textslider\_value2").innerHTML = slider\_value2; console.log(slider\_value2);

var xhr2 = new XMLHttpRequest();

xhr2.open("GET", "/slider?value2="+slider\_value2, true); xhr2.send();

}

</script>

</body>

</html>

)rawliteral";

// Przypisywanie wartości suwaka do zmiennej

String processor(const String& var){ if (var == "SLIDERVALUE"){

return slider\_value;

}else if(var=="SLIDERVALUE2"){ return slider\_value2;

}else{

return String();

}

}

//Zasada działania silników void setup(){ Serial.begin(115200); delay(1000); pinMode(ENA\_pin, OUTPUT); pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT); pinMode(ENB\_pin, OUTPUT); pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);

analogWrite(ENA\_pin, slider\_value.toInt()); digitalWrite(IN1, LOW);

digitalWrite(IN2, LOW); analogWrite(ENB\_pin, slider\_value2.toInt()); digitalWrite(IN3, LOW);

digitalWrite(IN4, LOW);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000); Serial.println("Connecting...");

}

Serial.println(WiFi.localIP());

//Łączenie oraz sprawdzanie uwieżytetylności do oprogramowania server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request){ request->send\_P(200, "text/html", index\_html, processor);

});

server.on("/slider", HTTP\_GET, [] (AsyncWebServerRequest \*request) { String message;

if (request->hasParam(input\_parameter)) {

message = request->getParam(input\_parameter)->value(); slider\_value = message;

analogWrite(ENA\_pin, slider\_value.toInt()); digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

}else if (request->hasParam(input\_parameter2)) {

message = request->getParam(input\_parameter2)->value(); slider\_value2 = message;

analogWrite(ENB\_pin, slider\_value2.toInt()); digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

}else{

message = "No message sent";

}

Serial.println(message);

request->send(200, "text/plain", "OK");

});

server.begin();

}

void loop() {

}

Powyższy kod przedstawia ogólne działanie strony

# Kod napisany na raspberry pi

import RPi.GPIO AS GPIO

import sys

import time

import signal

Button\_GPIO = 16

GPIO.setmode(GPIO.BCM)

GPIO.setup(BUTTON\_GPIO,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

# global variable

counter = 0

# constants

SAMPLING\_TIME = 1 # ( in seconds )

# interrupt handler function

def tachometer(pin):# pin is default positional argument

global counter

counter += 1

GPIO.ad\_event\_detect(BUTTON\_GPIO,GPIO.FALLING,

callback=tachometer,boucetime=100)

# attach the interrupt to the tachometer Pin

tachometerPin.irq(trigger = Pin.IRQ\_RISING,

handler = tachometer)

# main logic/ function of the program

while True:

utime.sleep(SAMPLING\_TIME)

revolutions\_per\_sampling\_time = counter / 2 # two white strips on the rotor

revolutions\_per\_sec = revolutions\_per\_sampling\_time / SAMPLING\_TIME

revolutions\_per\_minute = revolutions\_per\_sec \* 60

print("RPM : ", revolutions\_per\_minute )

# reset the counter to zero

counter = 0

Powyższy kod opisuje działanie raspberry pi i czujnika podczerwieni

# Wnioski

Wykonanie projektu linii produkcyjnej pozwoliło rozszerzyć naszą wiedzę z zakresu działania poszczególnych modułów w połączeniu z ESP8266.

Dzięki modułom ESP8266 można stworzyć system umożliwiający nam sterowanie zdalne oraz możliwość programowania rzeczy w obrębie sieci

domowej. Można je wykorzystać np. do tworzenia inteligentnego budynku lub też sterowania małymi silniczkami chociażby w ogrodach jako timer z pompą

lub też małej domowej linii produkcyjnej.

Dzięki raspberry pi można również otrzymać porównanie dzięki któremu można bezpośrednio mierzyć wartości maszyny bez bezpośredniego ingerowania w nią