| ASK 2021 | | | | |
| --- | --- | --- | --- | --- |
| Prowadzący | Wydział, kierunek | Semestr | Rok | Nr grupy |
| dr inż. Rafał Klaus | WIiT, informatyka | IV | 2 | 1024 |

LINE FOLLOWER

[Wprowadzenie](#_t5jjy7uo4rks) 3

[Specyfikacja wymagań](#_o3krexc0je37) 3

[Opis rozwiązania systemu komputerowego](#_1wchvrf6dou1) 3

[Krótki opis prac](#_4q2j2h2j3yb0) 4

[Test komponentów](#_33mub34018vn) 4

[Projektowanie płytek PCB](#_22j1tyng3) 4

[Budowa prototypu](#_ppnur03k5b9m) 4

[Przygotowanie laminatu](#_54zzk1t65gyf) 5

[Trawienie](#_ezrdoq70fz77) 5

[Montaż komponentów](#_yantl9ipsvu) 5

[Finalny projekt i jazda próbna](#_9ii28m3f0qh5) 6

[Opis konstrukcji](#_3c2c5mpc8mp9) 7

[Elementy](#_unceisxnlojn) 7

[Budowa](#_gwb6fisjjktf) 7

[Oprogramowanie](#_aesdmrx5z38k) 9

[Koncepcja programowa](#_p0sw1uspfful) 9

[Pomysł: Wykorzystanie Arduino](#_icr480yommg5) 9

[Arduino bez Arduino](#_atmleorc1onr) 9

[Przygotowanie Atmegi](#_beypb08vr612) 9

[Odpowiednie podłączenie](#_kl5klbpwxawj) 9

[Oprogramowanie Arduino](#_a2j822oxtq0x) 10

[Rezultat](#_8gqioq19ai4x) 10

[Programowanie mikrokontrolera Atmega za pomocą płytki Arduino](#_lv2tlm8ob929) 11

[Schemat połączeń](#_omx9t6ib0gir) 11

[Wgrywanie programu](#_jh5vyyhynryt) 12

[Program](#_ve2t9kv72j7e) 12

[Opis elektroniczny](#_k1lh574svxks) 17

[Opis ogólny](#_knraxhowk4ip) 17

[Transoptor odbiciowy CNY70](#_mixrcjovlxoe) 17

[Stabilizator napięcia (7805TV)](#_9dsgzax6884p) 17

[Mostek H (L293D)](#_z0gmr1acdhrd) 18

[Programator](#_9pbu2eft3ay5) 19

[Mikrokontroler Atmega (Atmega328P-U)](#_dpji70det04r) 20

[Schematy oraz opis](#_oc2h2i7jpze5) 22

[Schemat płytki z Atmegą oraz mostkiem H](#_kbkv2lfm94zm) 22

[Kosztorys](#_p2yy13q15w9e) 24

[Autorzy](#_wdgbgabkin5q) 25

# Wprowadzenie

## Specyfikacja wymagań

Zaplanowanym zadaniem było utworzenie robota typu line follower. Na podstawie odczytów z czujników świetlnych, robot miał za zadanie podążać wyznaczoną trasą bez zewnętrznego sterowania. Jego podstawowymi funkcjami miało być:

* Odczytywanie wartości z czujników
* Wyznaczanie kierunku jazdy na podstawie odczytanych wartości
* Uruchamianie silników w odpowiednich proporcjach mocy, w celu odpowiedniego nakierowania
* Bieżąca korekta trasy na podstawie danych podawanych w czasie rzeczywistym

Zaplanowane było utworzenie robota na własnoręcznie wytrawionych płytkach drukowanych, i zaprogramowanie go przy pomocy mikrokontrolera Atmega 328P-U. W celu ułatwienia możliwości wprowadzanie bieżących poprawek, zdecydowaliśmy się skorzystać z opcji "bootloadowania" danego mikrokontrolera przy pomocy posiadanego wcześniej kontrolera Arduino.

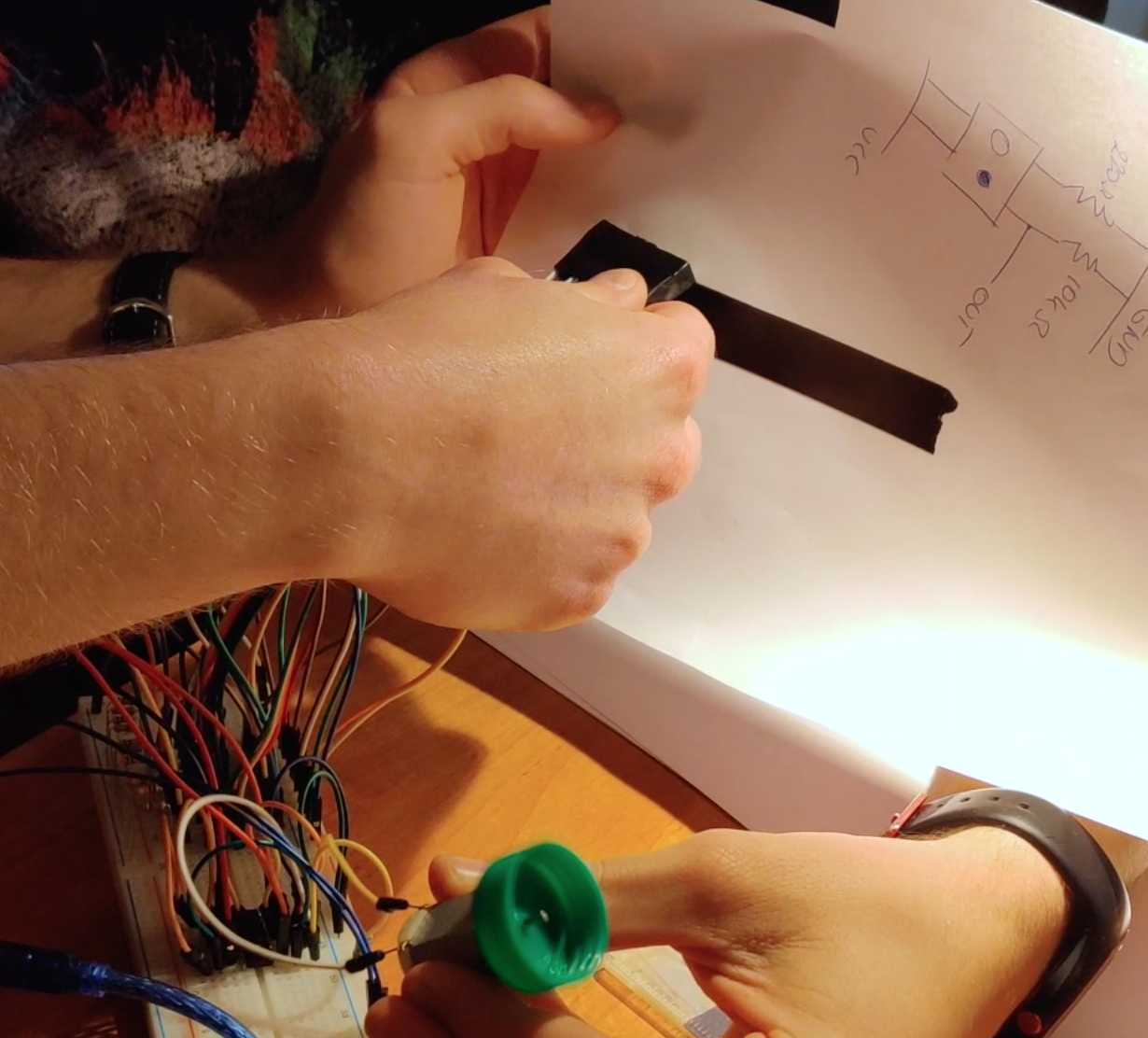
## Opis rozwiązania systemu komputerowego

Naszym pierwszym zadaniem było zaprogramowanie mikrokontrolera Atmega, w celu użycia go jako główny kontroler robota. W tym celu podłączyliśmy kontroler Arduino z wymontowanym wcześniej mikrochipem do komputera, a na jego miejsce podłączyliśmy Atmegę. Za co pomocą dedykowanej aplikacji wgraliśmy program rozruchowy, dzięki czemu robot mógł funkcjonować bez Arduino.

# Krótki opis prac

## Test komponentów

Konstrukcje naszego projektu zaczęliśmy od testowania czujników. Pięć czujników wraz z rezystorami oraz LED-ami zostały podłączone na płytce prototypowej. Następnie za pomocą Arduino pobraliśmy pomiary w różnych warunkach świetlnych, czujniki były skierowane na białą kartkę lub czarną taśmę. Każda lampka LED odpowiadała za jeden czujnik, światło oznaczało pomiar oświetlenia nad czarną taśmą, a brak światła pomiar nad kartką. Na tym etapie również zaczęło powstawać oprogramowanie, które później zostanie użyte przy finalnym projekcie.



Kolejnym etapem było połączenie czujników z silnikami (nakrętka pełniła funkcję koła), które zastąpiły prostą symulację LED-ami. Silniki (sterowane dzięki mostkowi H) odpowiednio reagowały na pomiary czujników według scenariusza:

* jazdy prosto (oba silniki uruchomione)
* łagodnego zakrętu w prawo/lewo (jeden z silników uruchomiony na wyższych obrotach niż drugi)
* ostrego zakrętu w prawo/lewo (jeden z silników uruchomiony, drugi wyłączony)

## Projektowanie płytek PCB

W trakcie testowania komponentów zaczęły powstawać projekty dwóch płytek PCB. Na pierwszej płytce zaplanowaliśmy umieszczenie czujników (wraz z rezystorami oraz wyprowadzeniami), a na drugiej mikrokontroler oraz mostek H. Projekty powstały w oprogramowaniu EAGLE. Schematy oraz ich opisy można znaleźć w poniższej sekcji.

## Budowa prototypu

Do zbudowania prototypu wykorzystaliśmy klocki marki “LEGO”. Obudowaliśmy wokół nich silniki z zamocowanymi kołami oraz przymocowaliśmy do przodu pojazdu czujniki. Na tak zbudowanej konstrukcji umocowaliśmy płytki prototypowe wraz z zasilaniem. Zadbaliśmy również o odpowiedni rozstaw kół tak, aby pojazd mógł bezproblemowo skręcać.

## Przygotowanie laminatu

Przygotowane uprzednio schematy wydrukowaliśmy na papierze kredowym. Następnie przycięliśmy laminaty miedziane do rozmiaru nieznacznie przekraczającego rozmiar schematów. Później zgrzewaliśmy laminat - przez pół godziny każdy - ze schematami za pomocą żelazka. Tak zgrzane laminaty ze schematami moczyliśmy w wodzie z płynem do naczyń przez 20 minut. Następnie odkleiliśmy papier od laminatu i podziwialiśmy przetransferowany toner. Braki w tonerze uzupełniliśmy markerem permanentnym. Aby odtłuścić laminaty wykorzystaliśmy aceton.

## Trawienie

Do trawienia wykorzystaliśmy nadsiarczan sodu - najpopularniejszy wytrawiacz. Przygotowaliśmy się do tej czynności bardzo dokładnie - założyliśmy maseczki, okulary ochronne oraz otworzyliśmy okno. Zmieszaliśmy pół litra wody w temperaturze 50°C z opakowaniem nadsiarczanu sodu. Tak przygotowany płyn rozlaliśmy do dwóch naczyń, w których umieściliśmy płytki. Trawienie trwało ok. 50 minut. Roztwór w tym czasie zmienił kolor na błękitny. Aby zmyć toner wykorzystaliśmy ponownie aceton.

## Montaż komponentów

Po przetestowaniu komponentów oraz pomyślnym wytrawieniu płytek przystąpiliśmy do finalnego łączenia części. Jednak zanim do tego doszło, należało poprawić płytki. Nie wszystkie otwory zostały wywiercone perfekcyjnie, przez co należało odpowiednio wydrapać miedź przy tych otworach tak, aby nie powstało zwarcie. Następnie przeszliśmy do przylutowania komponentów do płytek.

## Finalny projekt i jazda próbna

Zlutowane płytki z komponentami umieściliśmy w finalnym szkielecie pojazdu. Następnie zbudowaliśmy trasę dla naszego robota składającą się zakrętów po łuku, jednego zakrętu 90°, pętli oraz odcinków prostych. Funkcjonowanie pojazdu testowaliśmy na dwóch mocach silników: słabszej i mocniejszej. Różnice w zachowaniu można zauważyć na filmie.

# Opis konstrukcji

## Elementy

Główne elementy składowe robota:

* płytka główna

Na niej znajdują się najważniejsze komponenty - mikrokontroler wraz z podłączonym do niej mostkiem H. Na połączeniach znajdują się także rezystor 10K, kondensatory ceramiczne (100nF oraz 22pF), kondensatory elektrolityczne (10uF oraz 22uF), stabilizator napięcia, oraz rezonator kwarcowy 16MHz. Części te służą dodaniu funkcjonalności zwykłemu mikroprocesorowi Atmega, aby mógł on swoim działaniem w pełni zasymulować działanie kontrolera Arduino

* płytka z czujnikami i odpowiednimi rezystorami

Płytka, dzięki której robot wie jaki kierunek obrać. 5 czujników (wraz z rezystorami 10K i 220) ułożonych w kształt V na bieżąco przesyła informacje do płytki głównej. Dzięki odpowiedniemu oprogramowaniu, nasz mikroprocesor na podstawie wysłanych do niego danych określa kierunek jazdy.

* cztery baterie AA

Cztery baterie połączone szeregowo dostarczają odpowiednie napięcie do głównej płytki miedzianej. Dodatkowo stabilizator wraz z kondensatorami dba o równomierną i stałą w czasie pracę układu.

* bateria 9V

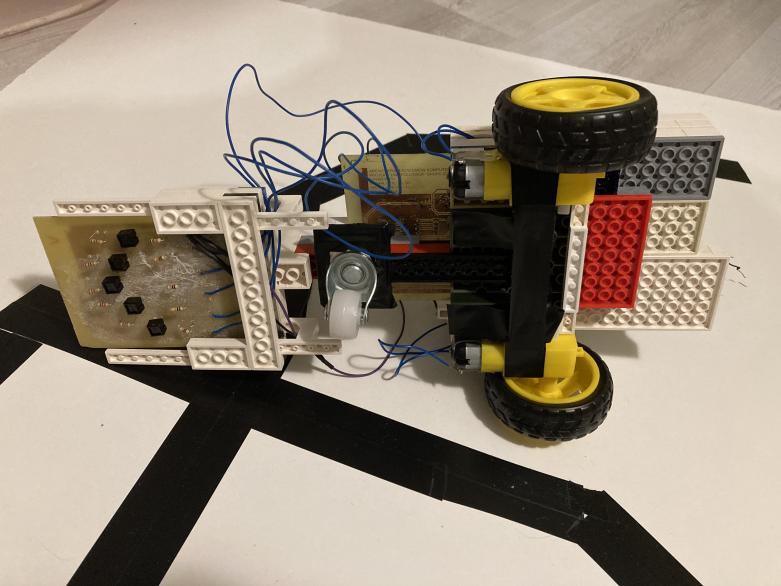
Bateria 9V zasila poprzez mostek H silniki odpowiedzialne za ruch. Osobne zasilanie niezależne od zasilania logiki układu pozwala na osiągnięcie wyższej mocy obrotowej silnika oraz niezależność mikroprocesora - zdecydowanie ułatwiało to pracę w przypadku szukania awarii.

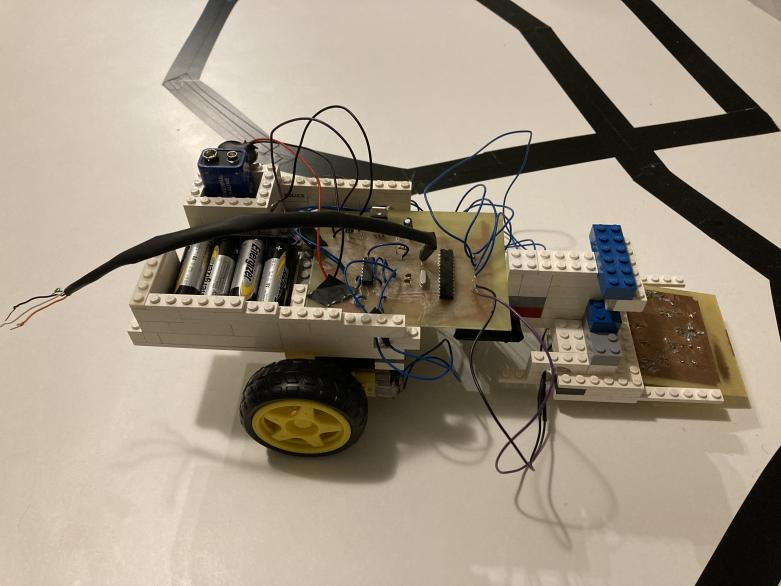
* silniki wraz z kołami

Główna oś napędowa robota - dwa silniki podłączone do mostka H na bieżąco otrzymują odpowiednią ilość mocy, by jak najsprawniej trzymać się wyznaczonej trasy.

## Budowa

Szkielet i obudowa robota złożona jest z klocków LEGO. Robot posiada dwie wytrawione płytki miedziane: jedna na czujniki, drugą na pozostałe komponenty (mikrokontroler (Atmega328P-U), mostek H), która połączona jest z silnikami. Do płytki głównej dochodzą dwa źródła zasilania - 4 baterie AA odpowiedzialne za logikę układu, oraz jedna bateria 9V dostarczająca przez mostek H moc do napędu. Płytka z czujnikami łączy się z główną płytką, na której mikrokontroler przetwarza odczyty czujników i na podstawie wgranego programu w języku C przekazuje sygnały do silników, na których zamocowane są koła o odpowiednio dostosowanym rozstawie. Trzecie koło zamontowane jest tuż za płytką z czujnikami w celu wspomagania skręcania i natychmiastowego reagowania na zmiany odczytów przez czujniki oraz stabilizacji robota. Dodatkowo, szerokość rozstawu kół odpowiada szerokości płytki, przez co robot nie jest zbyt szeroki, oraz jego ciężar jest w dużym stopniu zrównoważony. Główny punkt ciężkości robota znajduje się nad osią kół - wspomaga to stabilność pojazdu, dzięki czemu przy gwałtownych zwrotach ryzyko przewrócenia się robota jest zminimalizowane. Czujniki są umieszczone bardzo blisko ziemi, w celu dokładnych odczytów trasy, oraz zniwelowania wpływu zewnętrznego światła na ich odczyty.





# Oprogramowanie

## Koncepcja programowa

### Pomysł: Wykorzystanie Arduino

[[1]](#footnote-0)

Arduino to [platforma programistyczna](https://pl.wikipedia.org/wiki/Framework) dla [systemów wbudowanych](https://pl.wikipedia.org/wiki/System_wbudowany) oparta na prostym projekcie [Open Hardware](https://pl.wikipedia.org/wiki/Open_Hardware) przeznaczonym dla [mikrokontrolerów](https://pl.wikipedia.org/wiki/Mikrokontroler) montowanych w pojedynczym [obwodzie drukowanym](https://pl.wikipedia.org/wiki/Obw%C3%B3d_drukowany), z wbudowaną obsługą [układów wejścia/wyjścia](https://pl.wikipedia.org/wiki/Uk%C5%82ad_wej%C5%9Bcia-wyj%C5%9Bcia) oraz standaryzowanym [językiem programowania](https://pl.wikipedia.org/wiki/J%C4%99zyk_programowania)[[2]](#footnote-1). Powszechność tego systemu, jego dopracowanie, dostępność materiałów oraz dokumentacji sprawia że jest jednym z pierwszych wyborów podczas prostych projektów elektronicznych.

### Arduino bez Arduino

Oczywiście wykorzystanie gotowej płytki (np. Arduino UNO) wraz z gotowymi sterownikami komponentów jest dalece prostszym zadaniem niż przygotowanie własnego systemu komputerowego. W zasadzie, cały projekt ograniczyłby się wtedy do odpowiedniego połączenia kilku - kilkunastu portów w odpowiedniej konfiguracji.

Udało nam się zatem osiągnąć coś innego. Wykorzystaliśmy płytkę (oraz software) Arduino by zaprogramować mikroprocesor Atmega w taki sposób by “myślał” że jest płytką Arduino. Mówiąc mniej kolokwialnie - przyłączając do Atmegi kilka podstawowych części (rezystory, kondensatory, zegar kwarcowy) udało nam się załadować program rozruchowy (bootloader), który współpracował z oprogramowaniem Arduino. W ten sposób mogliśmy programować mikrokontroler Atmega wykorzystując oprogramowanie Arduino IDE (i wszystkie jego zalety), a w finalnym projekcie nie korzystać w ogóle z płytki Arduino.

## Przygotowanie Atmegi

### Odpowiednie podłączenie

Aby operacja przebiegła pomyślnie, należy zastosować połączenia zgodnie ze schematem[[3]](#footnote-2) poniżej. Elementy elektryczne:

* opornik o rezystancji 10 kiloOmów
* dwa kondensatory o pojemności 22 pikoFaradów
* rezonator kwarcowy (zegar) 16 MHz.
* mikroprocesor Atmega328P / Atmega168 / Atmega8 (opisana poniżej metoda powinna być zgodna dla wszystkich mikrokontrolerów z rodziny atmega w obudowie przewlekanej DIP)

### 

### Oprogramowanie Arduino

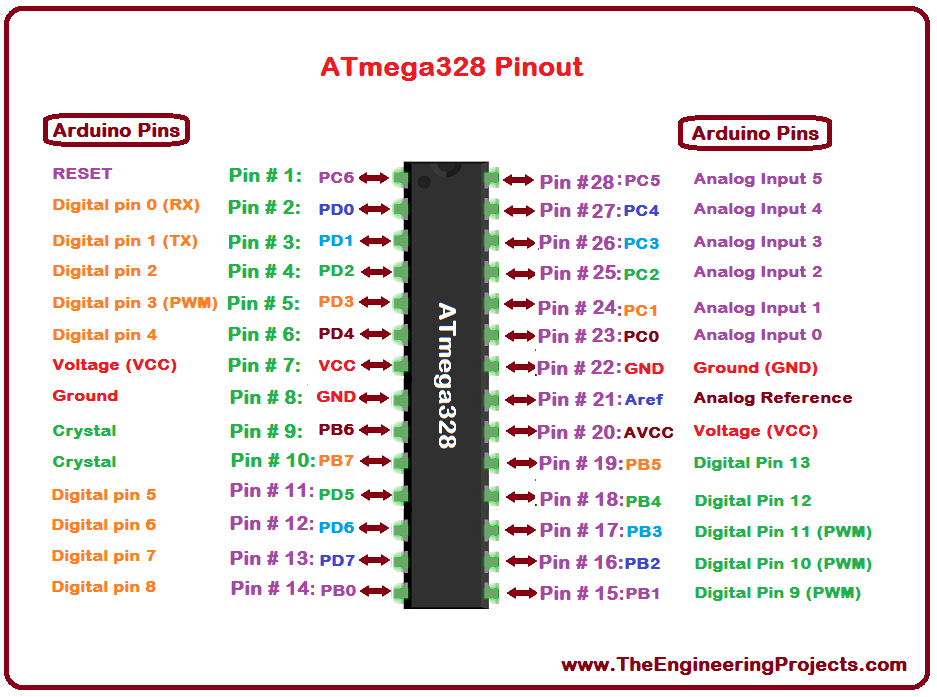
Tak przygotowany mikrokontroler należy podpiąć do komputera (poprzez port USB znajdujący się przy płytce Arduino, uruchomić oprogramowanie Arduino IDE i wykonać następujące kroki:

1. Załadować szkic “ArduinoISP” do płytki Arduino (Plik > Przykłady > ArduinoISP)
2. Wybrać odpowiednią płytkę (Narzędzia > Płytka). W naszym przypadku było to Arduino UNO
3. Wybrać programator “Arduino as ISP” (Narzędzia > Programator > Arduino as ISP)
4. Uruchomić wypalenie bootloader’a (Narzędzia > Wypal bootloader)

### Rezultat

Jeśli wszystkie połączenia były poprawne, operacja powinna zakończyć się sukcesem. Co zatem udało się osiągnąć? Wgrany program rozruchowy do naszego mikrokontrolera sprawia że jest on od tej chwili zgodny ze standardem Arduino, co oznacza że można go wpiąć do innej płytki Arduino (i będzie działać) albo używać go niezależnie, ale mieć możliwość programowania go za pomocą oprogramowania ArduinoIDE.

Oczywiście pojawia się problem: może i tak przygotowana Atmega technicznie zachowuje się jak płytka Arduino, jednak jak jej używać? Które piny odpowiadają którym portom? Z pomocą przychodzi nam poniższy schemat[[4]](#footnote-3):



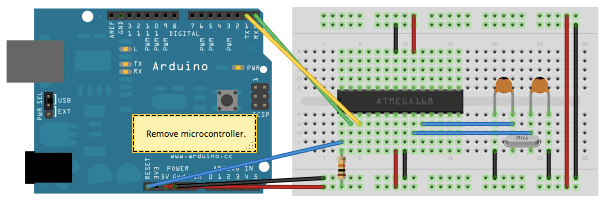
Korzystając z powyższego porównania można z łatwością przenieść myślenie o płytce Arduino na niezależny mikroprocesor.

## 

## Programowanie mikrokontrolera Atmega za pomocą płytki Arduino

### Schemat połączeń

Jeżeli mikrokontroler Atmega który chcemy zaprogramować znajduje się poza płytką Arduino, nie jest to dużym problemem. Możemy wykorzystać do zaprogramowania go naszą płytkę Arduino. **Ważne: do zastosowania tej metody należy wyciągnąć fabryczny mikroprocesor Atmega z Arduino!** Schemat[[5]](#footnote-4) połączeń prezentuje się następująco:



Warto zauważyć że już na tym etapie praca z wcześniej przygotowanym mikroprocesorem staje się dość prosta i przyjemna. Wystarczy połączyć (“na krzyż”) porty RX i TX (odpowiedzialne za przesył danych) pomiędzy Atmegą a płytką Arduino, porty resetu a także napięcie oraz uziemienie ze sobą i to wszystko.

### Wgrywanie programu

Wgranie programu na nasz mikroprocesor Atmega odbywa się teraz w dokładnie tak jakbyśmy korzystali ze standardowej płytki Arduino. W *Narzędziach* wybieramy płytkę (w naszym przypadku Arduino UNO), a także programator *AVRISP mkII*, klikamy *wgraj* i gotowe! Mikroprocesor od teraz będzie przetwarzał sygnały zgodnie z wgranym programem.

## Program

Deklaracja zmiennych dla silników oraz czujników

| //Motor A (right) const int rightBck = 11; const int rightFwd = 10; int rightBckValue = 0; int rightFwdValue = 0;  //Motor (left) const int leftBck = 9; const int leftFwd = 3; int leftBckValue = 0; int leftFwdValue = 0;  const int slower = 160; const int faster = 200;  // Sensors const int sensorPin[5] = {A0, A1, A2, A3, A4};  int sensorValue[5]; |
| --- |

Wstępny setup programu. Ustawienie zadeklarowanych wcześniej pinów (odpowiadających za silniki) jako wyjście oraz początkowe wyzerowanie mocy silników.

| //This will run only one time. void setup(){  Serial.begin(9600);  //Set pins as outputs  pinMode(rightBck, OUTPUT);  pinMode(rightFwd, OUTPUT);  pinMode(leftBck, OUTPUT);  pinMode(leftFwd, OUTPUT);    //And this code will stop motors  digitalWrite(rightBck, LOW);  digitalWrite(rightFwd, LOW);  digitalWrite(leftBck, LOW);  digitalWrite(leftFwd, LOW); } |
| --- |

Zczytywanie wartości sensorów oraz deklaracja prostej funkcji, dzięki której mogliśmy obserwować pomiary czujników w konsoli Arduino IDE.

| void readSensors() {  for (int i = 0; i < 5; i++) {  sensorValue[i] = analogRead(sensorPin[i]);  } }  void printSensors() {  for (int i = 0; i < 5; i++) {  Serial.print("A");  Serial.print(i);  Serial.print(": ");  Serial.println(sensorValue[i]);  }  Serial.println("\*\*\*\*\*\*\*");  Serial.println(rightFwdValue);  Serial.println(leftFwdValue); } |
| --- |

Funkcja wysyłająca odpowiednie wartości (zależne od pomiarów czujników) do pinów odpowiadających lewemu i prawemu silnikowi.

| void drive() {  analogWrite(rightBck, rightBckValue);  analogWrite(rightFwd, rightFwdValue);  analogWrite(leftBck, leftBckValue);  analogWrite(leftFwd, leftFwdValue); } |
| --- |

Funkcja obliczająca, które czujniki wykryły linię (wartość true). Jazda jest podzielona na segmenty: jazda na wprost, lekko w lewo/prawo (jedno z kół kręci się na wolnych obrotach, drugie z pełną mocą), mocno w lewo/prawo (jedno z kół kręci się z pełną mocą, drugie nie kręci się) oraz cofanie - w przypadku, gdy robot nie wykrywa linii (wszystkie wartości tablicy sbv to false).

| void calculateValues() {  // A0 - A1 - A2 - A3 - A4  // Calculating boolean values (path detected  // or not detected) for simpler calculations   // SBV is an acronym for "Sensor Bool Value"    bool sbv[5];  for (int i = 0; i < 5; i++) {  if (sensorValue[i] < 100) {  sbv[i] = true;  }  else {  sbv[i] = false;  }  }    // Drive forward if corresponding sensors on  // left / right are equal and at least one of  // A0, A1, A2 is true;   if (sbv[1] == sbv[3] && sbv[0] == sbv[4] && (sbv[0] || sbv[1] || sbv[2])) {  Serial.println("Driving Forward");  rightFwdValue = slower;  leftFwdValue = slower;  rightBckValue = 0;  leftBckValue = 0;  }   // Turn right (slightly) if A3 is true and  // A0, A1, A4 are false   else if (sbv[3] && !sbv[0] && !sbv[1] && !sbv[4]) {  Serial.println("Turning right (slightly)");  rightFwdValue = slower;  leftFwdValue = faster;  rightBckValue = 0;  leftBckValue = 0;  }   // Turn right (narrow) if A4 is true and  // A0, A1 are false   else if (sbv[4] && !sbv[0] && !sbv[1]) {  Serial.println("Turning right (narrow)");  rightFwdValue = 0;  leftFwdValue = faster;  rightBckValue = 0;  leftBckValue = 0;  }   // Turn left (slightly) if A1 is true and  // A0, A3, A4 are false   else if (sbv[1] && !sbv[0] && !sbv[3] && !sbv[4]) {  Serial.println("Turning left (slightly)");  rightFwdValue = faster;  leftFwdValue = slower;  rightBckValue = 0;  leftBckValue = 0;  }   // Turn left (narrow) if A0 is true and  // A3, A4 are false   else if (sbv[0] && !sbv[3] && !sbv[4]) {  Serial.println("Turning left (narrow)");  rightFwdValue = faster;  leftFwdValue = 0;  rightBckValue = 0;  leftBckValue = 0;  }   else if (!sbv[0] && !sbv[1] && !sbv[2] && !sbv[3] && !sbv[4]) {  Serial.println("Reverse");  rightFwdValue = 0;  leftFwdValue = 0;   rightBckValue = slower;  leftBckValue = slower;    drive();  delay(100);    rightBckValue = 0;  leftBckValue = 0;  }  else {  rightFwdValue = slower;  leftFwdValue = slower;  } } |
| --- |

Pętla główna programu, czyli pomiar wartości z czujników, drukowanie wyników do konsoli, wybór odpowiedniego scenariusza jazdy (prosto/lewo/prawo) oraz wysłanie odpowiednich sygnałów do pinów połączonych z dwoma silnikami. Po 60 milisekundowym opóźnieniu silniki zostają wyłączone, a po kolejnym 10 milisekundowym opóźnieniu pętla się powtarza.

| void loop(){  readSensors();  printSensors();  calculateValues();  drive();    delay(60);   leftFwdValue = 0;  rightFwdValue = 0;  leftBckValue = 0;  leftFwdValue = 0;  drive();  delay(10); } |
| --- |

# Opis elektroniczny

## Opis ogólny

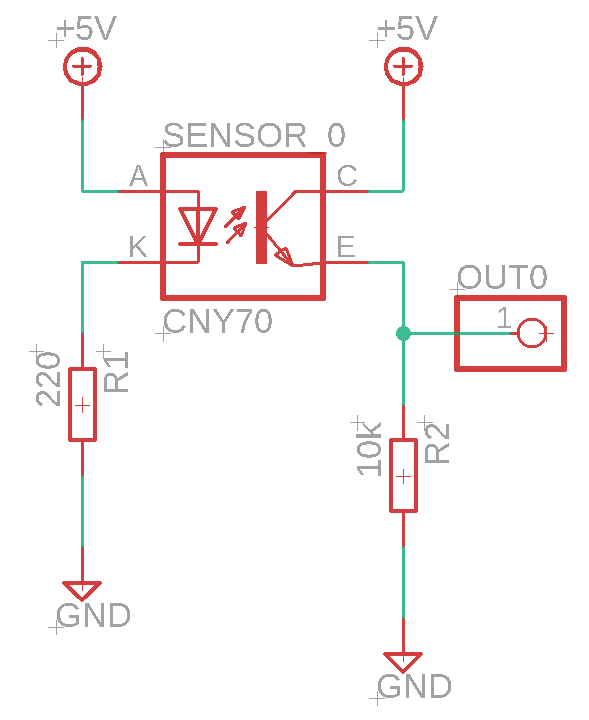
Nasz układ elektroniczny składa się z dwóch płytek drukowanych. Jedna z nich odpowiada za część logiczną. To tam znajduje się mikrokontroler Atmega, mostek H. To do tej płytki trafiają informacje zebrane przez czujniki, są przetwarzane by następnie podjąć decyzję o uruchomieniu danego silnika. Na drugiej z płytek znajdują się tylko i wyłącznie czujniki których zadaniem jest zbieranie informacji na temat położenia robota względem ścieżki.

## Transoptor odbiciowy CNY70

Pięć czujników CNY70 odpowiada za komunikację ze światem zewnętrznym. Znajdują się one na osobnej płytce, zasilanie pobierają z płytki głównej.

* PIN #A - zasilanie (anoda diody)
* PIN #K - GND (katoda diody)
* PIN #C - zasilanie
* PIN #E - GND

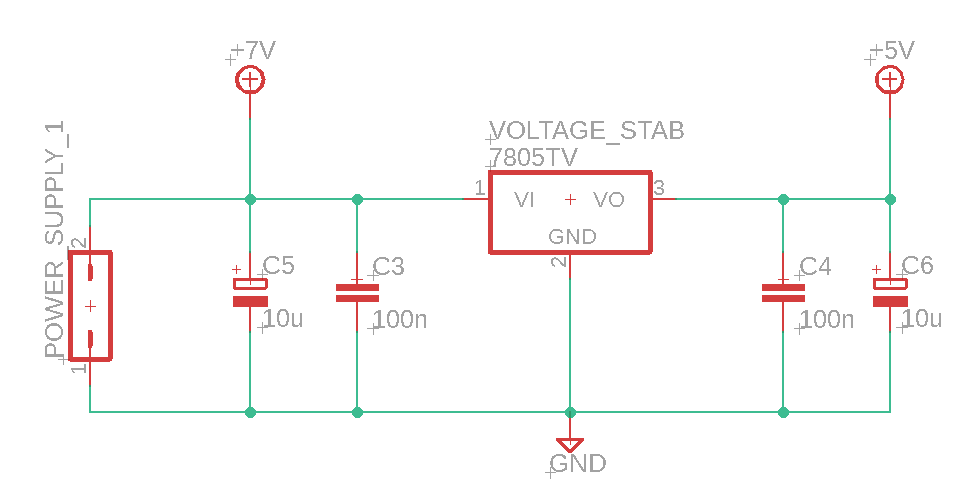
Dodatkowo PIN #E, z którego wychodzi sygnał informujący o odbiciu / nie odbiciu światła połączony jest z mikrokontrolerem Atmega.



## Stabilizator napięcia (7805TV)

Zalecane przez producentów napięcie na którym pracują mikrokontroler Atmega jak i mostek H wynosi 5V. Zasilanie części logicznej w naszym układzie składa się z 4 baterii AA, które łącznie generują napięcie około 6-7V. Stąd też pojawiła się potrzeba wykorzystania stabilizatora napięcia. Schemat połączenia, wraz z wymaganymi elementami biernymi zaprezentowany jest poniżej.

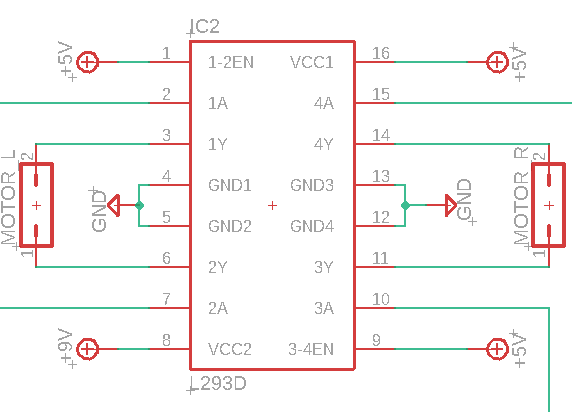
* PIN #1 - napięcie wejściowe (5 - 35V, w naszym przypadku około 6.3V)
* PIN #2 - GND
* PIN #3 - napięcie wyjściowe (5V)



## Mostek H (L293D)

Mostek H, który odpowiada za sterowanie silnikami posiada dwa źródła zasilania. Jedno, które odpowiada za część logiczną i jedno które związane jest bezpośrednio z silnikami.

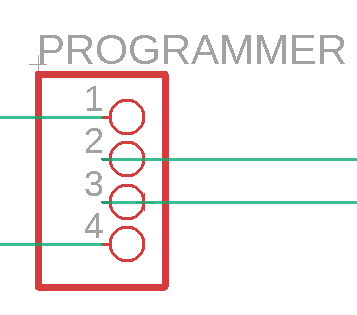
* PIN #1 - włączenie silnika 1 (lewego)
* PIN #2 - wejście sterujące nr 1 (silnik lewy)
* PIN #3 - wyjście sterujące nr 1 (silnik lewy)
* PIN #4, PIN #5, PIN #12, PIN #13 - GND
* PIN #6 - wyjście sterujące nr 2 (silnik lewy)
* PIN #7 - wejście sterujące nr 2 (silnik lewy)
* PIN #8 - zasilanie silników (9V)
* PIN #9 - włączenie silnika 2 (prawego)
* PIN #10 - wejście sterujące nr 3 (silnik prawy)
* PIN #11 - wyjście sterujące nr 3 (silnik prawy)
* PIN #14 - wyjście sterujące nr 4 (silnik prawy)
* PIN #15 - wejście sterujące nr 4 (silnik prawy)
* PIN #16 - zasilanie logiki (5V)



## Programator

Zakładając możliwość modyfikacji programu już po przylutowaniu atmegi, zarezerwowaliśmy 4 piny niezbędne do zaprogramowania urządzenia. Są to wcześniej omawiane piny RX, TX, RESET oraz GND. Opisany tutaj programator to tak naprawdę 4 połączenia wyprowadzone z Atmegi. Na zdjęciach naszego robota są one widoczne w postaci *antenki*.

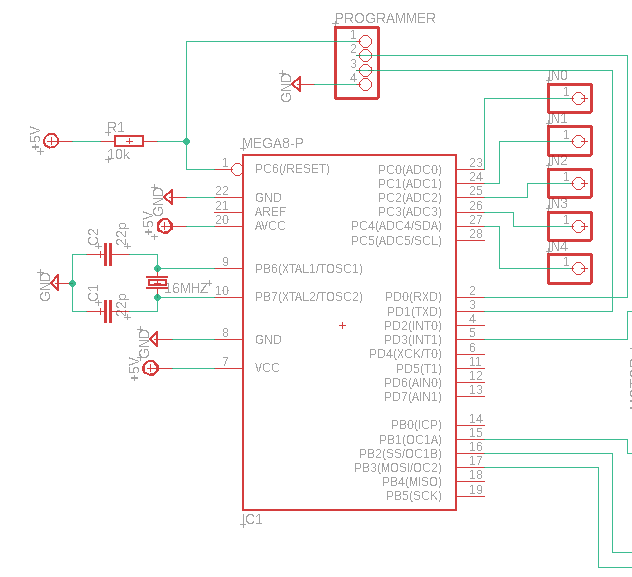
* PIN #1 - połączony z PIN #1 mikrokontrolera Atmega (RESET)
* PIN #2 - połączony z PIN #2 mikrokontrolera Atmega (RX)
* PIN #3 - połączony z PIN #3 mikrokontrolera Atmega (TX)
* PIN #4 - połączony z polem masy (GND)



## Mikrokontroler Atmega (Atmega328P-U)

Sercem (i mózgiem) naszego układu jest mikrokontroler Atmega328P - U. Łączy on wszystkie wyżej wymienione urządzenia i wprawia robota w ruch.

* PIN #1, PIN #2, PIN #3 - są to złącza *programatora*
* PIN #4 - nieaktywny
* PIN #5 - połączony z PIN #2 na mostku H (wejście sterujące nr 1)
* PIN #6 - nieaktywny
* PIN #7 - zasilanie (5V)
* PIN #8 - GND
* PIN #9, PIN #10 - rezonator kwarcowy
* PIN #11, PIN #12, PIN #13, PIN #14 - nieaktywne
* PIN #15 - połączony z PIN #7 na mostku H (wejście sterujące nr 2)
* PIN #16 - połączony z PIN #10 na mostku H (wejście sterujące nr 3)
* PIN #17 - połączony z PIN #15 na mostku H (wejście sterujące nr 4)
* PIN #18, PIN #19 - nieaktywne
* PIN #20 - zasilanie (5V)
* PIN #21 - nieaktywny
* PIN #22 - GND
* PIN #23, PIN #24, PIN #25, PIN #26, PIN #27 - połączone z odpowiednimi wyjściami czujników z drugiej płytki, to tymi połączeniami Atmega komunikuję się ze światem zewnętrznym
* PIN #28 - nieaktywny

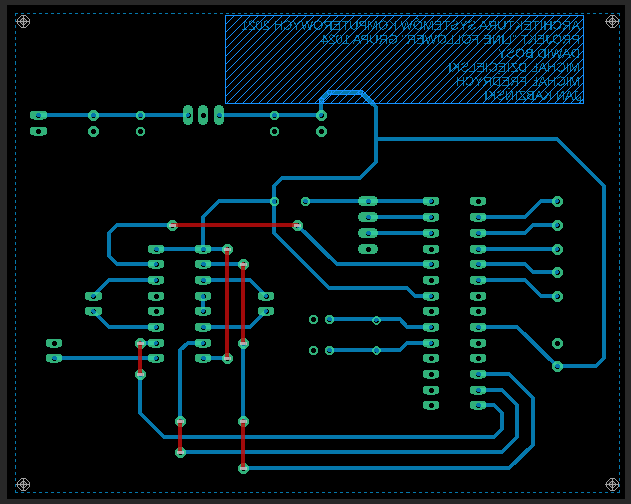


# Schematy i płytki

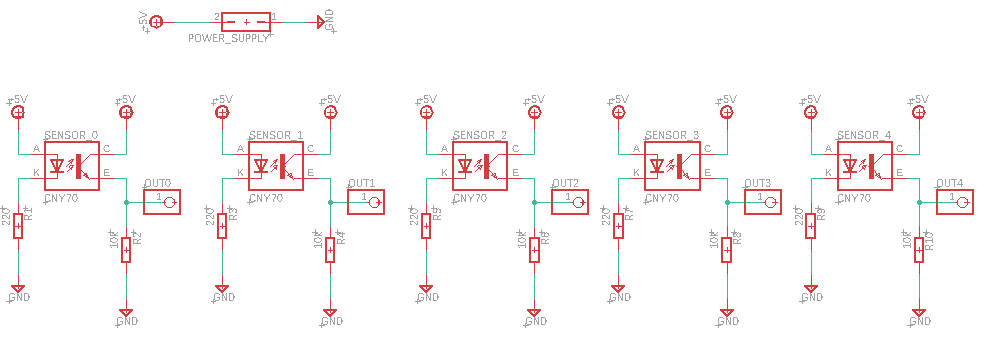
## Schemat płytki z Atmegą oraz mostkiem H (płytka główna)

## 

## Płytka z Atmegą oraz mostkiem H (płytka główna)



## Schemat płytki z czujnikami



## Płytka z czujnikami



# Kosztorys

| Część | Ile sztuk | Gdzie kupiona | Cena (zł) |
| --- | --- | --- | --- |
| Atmega 328P-U  Transoptor odbić  Stabilizator napięciowy 7805 5V/1.5A  Rezonator kwarcowy 16MHz HC49/U70  Rezystor węglowy 0.25W 10KΩ  Rezystor węglowy 0.25W 20KΩ  L293D - dwukanałowy sterownik silników 36V/0,6A  Kondensator ceramiczny 100nF/50V THT  Kondensator elektrolityczny 22uF/50V 5x11mm 105C THT  Kondensator elektrolityczny 10uF/50V 5x11mm 105C THT Rezystor THT CF węglowy 1/4W 10k - PAS-03967 Kondensator ceramiczny 22pF/50V THT  Wytrawiacz B327 - 100g CHE-01057  Laminat FR4 jednostronny rozmiar M LUT-00932  Dioda LED 5mm zielona LED-11144  Moduł zasilający do płytek stykowych MB102 - 3,3V 5V  Mini silnik MT78 3-6 V MOT-00951  Mocowanie do silników - MT78 NSZ-09786  Koszyk na 4 baterie typu AA (R6) bez przewodów  Koło + silnik 65x26mm 5V z przekładnią 48:1  Obrotowe kółko do podwozia robota  Klip na baterię 9V (6F22)  Dostawa | 1  5  3  1  100  100  5  10  10  10  30  10  2  4  10  1  3  3  2  2  2  2  2 | botland.com.pl | 9.50  2.90  2.90  1.20  8.00  8.00  8.90  0.99  1.50  1.00  1.90  0.99  8.80  18.00  1.50  5.90  15.00  2.85  2.80  19.80  3.00  1.50  25.80 |
| Arduino Uno  Marker permanentny  Bateria 9V  Bateria AA | 1  1  2  4 | Własne zasoby |  |
| Suma | | | 152.73 |

# Autorzy

| Autorzy | Indeks | Nr grupy dziekańskiej | Zadania |
| --- | --- | --- | --- |
| Dawid Bosy | 145396 | 1 | • Budowa algorytmu  • Lutowanie  • Tworzenie schematów  • Tworzenie filmu i dokumentacji |
| Michał Dzięcielski | 145323 | 1 | • Budowa algorytmu  • Budowa prototypu  • Przygotowanie płytek pod lutowanie  • Trawienie płytki  • Tworzenie filmu i dokumentacji |
| Michał Fredrych | 145269 | 1 | • Budowa algorytmu  • Lutowanie  • Przygotowanie płytek pod lutowanie  • Trawienie płytki  • Tworzenie filmu i dokumentacji  • Tworzenie schematów |
| Jan Kabziński | 145220 | 1 | • Budowa algorytmu  • Budowa prototypu  • Przygotowanie płytek pod lutowanie  • Tworzenie filmu i dokumentacji  • Trawienie płytki |

1. https://pl.wikipedia.org/wiki/Arduino [↑](#footnote-ref-0)
2. https://pl.wikipedia.org/wiki/Arduino [↑](#footnote-ref-1)
3. https://www.arduino.cc/en/Tutorial/BuiltInExamples/ArduinoToBreadboard [↑](#footnote-ref-2)
4. https://www.theengineeringprojects.com/wp-content/uploads/2017/07/ATmega328-Pinout.png [↑](#footnote-ref-3)
5. https://www.arduino.cc/en/Tutorial/BuiltInExamples/ArduinoToBreadboard [↑](#footnote-ref-4)