Łazik Lego

W oparciu o układ ATMega 2560

Autor: Patryk Reszczyński

Spis treści

Wstęp

Powyższy projekt ma za zadanie połączyć popularne na całym świecie klocki LEGO z układem ATMega 2560 na płytce Arduino Mega w celu skonstruowania automatu jeżdżącego. Jedną z koncepcji, na której opiera się projekt był wydany przez firmę LEGO robot LEGO Mindstroms. Umożliwia on składanie z gotowych części połączonych z „inteligentną kostką" robotów. Sercem tego urządzenia jest procesor oparty o architekturę ARM7 a w najnowszej wersji ARM9. Drugą inspiracją były łaziki zespołu Hyperion Team przeznaczone do wykonywania misji na obcych ciałach niebieskich. Robot Hyperion skonstruowany przez studentów Wydziału Mechaniki Politechniki Białostockiej został zwycięzca konkursu University Rover Challenge (URC) w roku 2013. W roku 2014 zwycięskim robotem była jego poprawiona wersja Hyperion 2. Trzecią inspiracją były roboty konstruowane w ramach zawodów walk robotów. Urządzenia przeznaczone do spychania lub niszczenia konkurencyjnych konstruktów z areny.

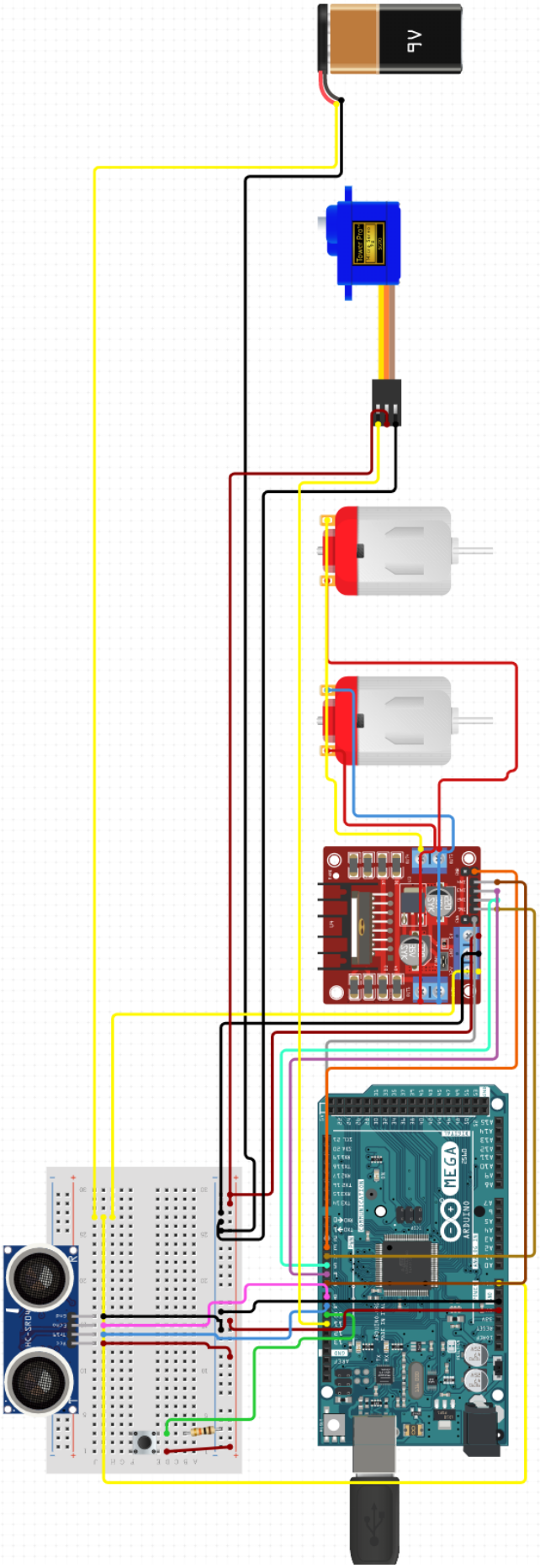
Cele

1. Skonstruowanie szkieletu robota pozwalająca na bezproblemowe przemieszczanie się
2. Implementacja napędu dzięki silnikom DC
3. Implementacja sterowania z pomocą układu serwomechanizmu
4. Implementacja wykrywania przeszkód dzięki ultradźwiękowemu czujnikowi odległości
5. Implementacja przycisku odpowiadającego za zapłon
6. Zaprogramowanie układu pozwalającego na jazdę oraz sprawne omijanie przeszkód i odnajdowanie ścieżek.

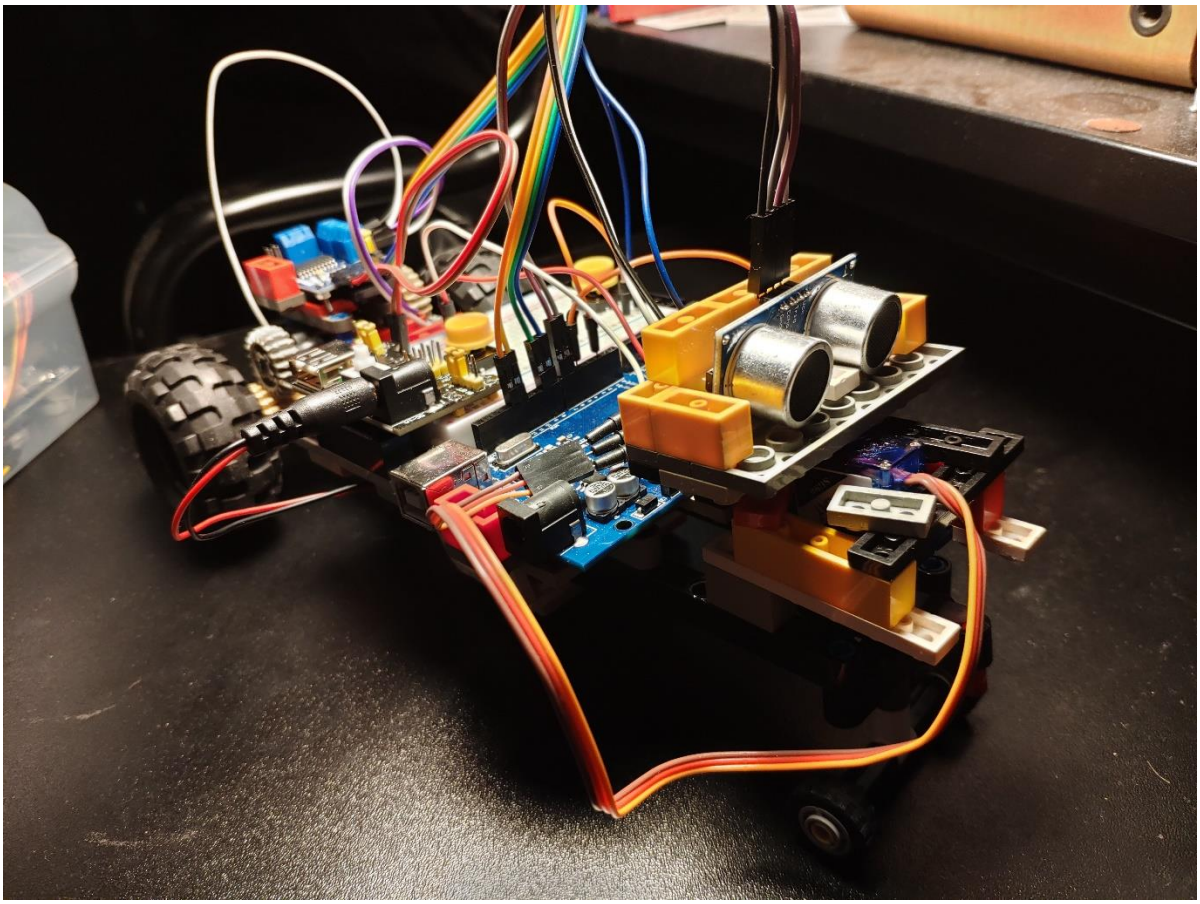
Elementy układu

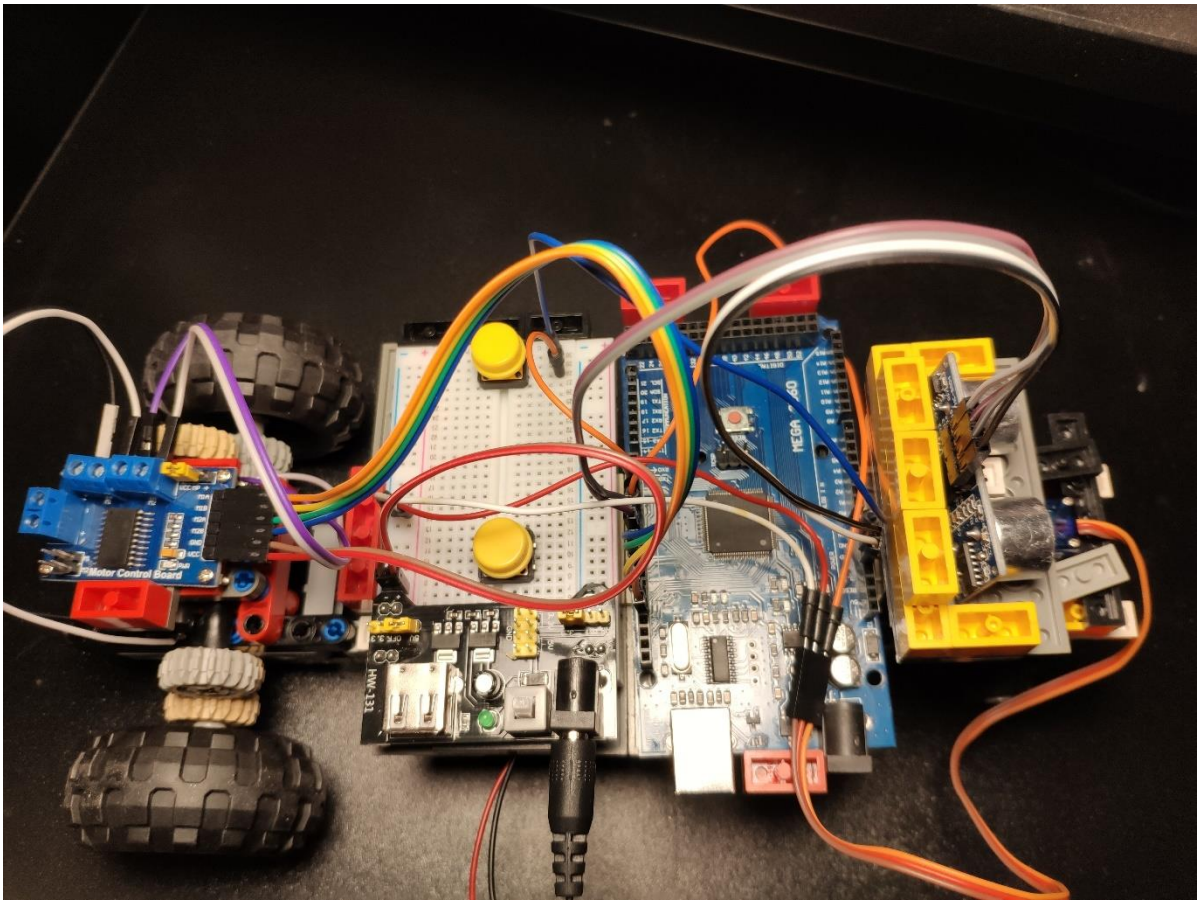
- Arduino ATMega 2560 lub zamiennik
- Płytka stykowa
- Moduł zasilający do płytek stykowych MB102 - 3,3V/5V
- Przycisk Tack Switch
- 2x Silnik N20-BT04 micro 30:1 1000RPM - 12V
- Sterownik silników Motor Control Board L293D - 36V/0,6A
- Serwo TowerPro SG-90
- Klocki Lego Classic
- Klocki Lego Technic
- 2x Adapter Lego – wał 3mm

Schemat



Zdjęcia projektu





Opis programu

Kod

Pełny kod programu dostępny jest na repozytorium GitHub autora pod adresem:

HTTPS://GITHUB.COM/RESZCZYK/ARDUINOLEGOROBOT/BLOB/MASTER/ROBOT.INO

Wykorzystane biblioteki

W projekcie wykorzystane dwie biblioteki:

1) NewPing.h – dzięki niej będzie można skorzystać z predefiniowanych metod odczytu i konwersji odległości na centymetry
2) Servo.h – biblioteka pozwalająca na sterowanie serwomechanizmem

Funkcja setup()

```
void setup() {
  Serial.begin(9600);
  pinMode(BUTTON, INPUT_PULLUP);

  pinMode(motor1_dir,OUTPUT);
  pinMode(motor1_pwm,OUTPUT);
  pinMode(motor2_dir,OUTPUT);
  pinMode(motor2_pwm,OUTPUT);

  servo.attach(servoPin);
}
```

W pierwszej linijce inicjalizujemy połączenie szeregowe do trybu serwisowego dzięki któremu będzie można odczytać poprawność odczytywanej przez czujnik zbliżeniowy odległości. Następnie inicjalizujemy przycisk z wbudowanym w układ rezystorem podciągającym do uruchomienia zapłonu silników. W kolejnych 4 linijkach podajemy piny odpowiedzialne za silniki DC. MotorX_dir odpowiada za kierunek obrotów (LOW/HIGH). MotorX_pwm odpowiada za prędkość obrotów. Ostatecznie jest przypisywany pin serwomechanizmu

Funkcje sterujące

Sterowanie napędem

```
void RunForward()
{
  digitalWrite(motor1_dir,HIGH);
  analogWrite(motor1_pwm,0);
  digitalWrite(motor2_dir,HIGH);
  analogWrite(motor2_pwm,0);
}

void RunBackward()
{
  digitalWrite(motor1_dir,LOW);
  analogWrite(motor1_pwm,255);
  digitalWrite(motor2_dir,LOW);
  analogWrite(motor2_pwm,255);
}

void StopMotors()
{
  digitalWrite(motor1_dir,LOW);
  analogWrite(motor1_pwm,0);
  digitalWrite(motor2_dir,LOW);
  analogWrite(motor2_pwm,0);
}
```

Za napęd odpowiadają dwa silniki DC. Powyższa funkcja RunForward() ustawia stan obu silników na HIGH i przesyła wartość 0 do pinu pwm co skutkuje pracą silników z maksymalną mocą. Analogicznie działa cofanie, funkcja RunBackward(). Zostaje w niej ustawiony stan LOW i maksymalna wartość dla silników 255. Funkcja StopMotors() zatrzymuje silnik ustawiając stan na LOW i przesyłając minimalną wartość 0.

Sterowanie skręcaniem

```
void ServoFront()
{
  delay(20);
  servo.write(80);
  delay(20);
}

void ServoLeft()
{
  delay(20);
  servo.write(50);
  delay(20);
}

void ServoRight()
{
  delay(20);
  servo.write(110);
  delay(20);
}
```

Za skręcanie łazika odpowiada serwomechanizm połączony z kołami przednimi. Do sterowania skręcaniem wykorzystane są 3 powyższe funkcje. Każda z nich ustawia pozycje serva w odpowiednim kierunku skręcania.

Pętla główna

```
void loop() {
  PowerOn();
  RunMotor();
}
```

W funkcji loop() znajdują się dwie funkcje. Pierwsza z nich PowerOn() odczytuje stan przycisku. Kiedy przycisk zostanie wciśnięty serwomechanizm zostaje ustawiony na pozycję centralną a zmienna POWER zmienia stan na true.

```
void PowerOn()
{
  if (digitalRead(BUTTON) == LOW && POWER == false)
  {
      ServoFront();
      POWER = true;
      delay(20);
  }
}
```

Po zmianie stanu POWER na true następuje zapłon silnika w funkcji RunMotor()

Funkcja jazdy łazika

```
void RunMotor()
{
  if (POWER == false)
  {
    StopMotors();
  }
  else if (POWER == true)
  {
      if (CheckIfObstacle() == true)
      {
        BypassObstacle();
      }
      else if (CheckIfObstacle() == false)
      {
        ServoFront();
        RunForward();
      }
   }
}
```

Funkcja RunMotor() najpierw sprawdza czy nastąpił zapłon (POWER == true). Jeśli nastąpił zapłon pętla sprawdza czy jest możliwość swobodnej jazdy przed siebie dzięki funkcji wyszukujących przeszkód [CheckIfObstacle()]. Jeśli nie ma żadnych przeszkód następuje wycentrowanie steru [ServoFront()] i jazda przed siebie [RunForward()] aż do napotkania przeszkody. Jeśli przeszkoda zostanie napotkana uruchomi się funkcja omijania przeszkód BypassObstacle().

Funkcje sprawdzają odległość

```
bool CheckIfObstacle()
{
  if (frontSonar.ping_cm() <= 30 && frontSonar.ping_cm() != 0 )
  {
    return true;
  }
  else if (frontSonar.ping_cm() > 30 || frontSonar.ping_cm() == 0)
  {
    return false;
  }
}
```

Funkcja sprawdza czy przed łazikiem znajduje się jakaś przeszkoda w odległości mniejszej lub równej 30 centymetrów. Jeśli sonar wykryje przeszkodę zostanie zwrócona wartość true (znaleziono przeszkodę). Kiedy do najbliższej przeszkody jest więcej niż 30 cm, funkcja zwraca false (brak przeszkody). Ze względu na wyniki podawane przez czujnik odległości, jeśli przeszkoda znajduje się dalej niż zakres pracy sonaru, odbierana jest wartość 0. Trzeba było wziąć tę wartość pod uwagę przy odczytywaniu przeszkód, ponieważ brak przeszkody mógłby oznaczać zwracanie wartości true przez to, że 0 < 30.

Podsumowanie

Powyższy projekt pozwolił mi na lepsze zapoznanie się z obsługą peryferii dostępnych dla Arduino takich jak silniki, serwomechanizmy i czujniki odległości. Wykorzystanie części mechanicznych wymagało wzięcia pod uwagę zwiększonego poboru prądu. Aby pozwolić na zasilenie wszystkich układu wykorzystano w projekcie baterię 9V podłączoną do specjalnego modułu zasilającego podłączonego do płytki stykowej. Mimo wszystko wymagane jest dodatkowe zasilanie dla płytki Arduino Mega co przysporzyło w trakcie testów niemało kłopotów. Ze względu na niedobory mocy funkcje łazika nie działały w prawidłowy sposób. Spowolniło to proces konstrukcyjny.

Wykorzystanie do konstrukcji klocków LEGO przyniosło dużo zabawy, ale też znacząco opóźniło proces projektowania. Połączenie silników z kołami łazika okazało się problematycznym procesem. Zbudowanie odpowiedniej konstrukcji pozwalającej na osadzenie silników napędzające koła okazało się trudniejsze niż we wstępnych założeniach i zajęło najwięcej czasu. Po wielu podejściach do modyfikacji konstrukcji, metodą prób i błędów udało się dojść do satysfakcjonującej budowy.

Konstrukcja łazika pozwala na późniejszą rozbudowę o dodatkowe funkcję takie jak:
- Dołożenie kolejnego czujnika zbliżeniowego do wykrywania przeszkód z tyłu urządzenia
- Dodanie ekranu wyświetlającego prędkość i odległość od przeszkód
- Dodanie modułu Bluetooth pozwalającego na sterowanie łazikiem za pomocą smartfonu

Cały proces projektowy i konstrukcyjny był dużym wyzwaniem. Trzeba było odświeżyć zagadnienia związane z elektroniką i mechaniką. Dzięki dokumentacji dostępnej w Internecie budowa łazika była ułatwiona a cały proces przyjemny i uczący. Przez charakterystykę Arduino duży nacisk położony został na programowaniu urządzenia w języku przyjemnym użytkownikowi.

Efekt finalny uważam za zadowalający i satysfakcjonujący.