ZESPÓŁ SZKÓŁ NR 4 W JAŚLE

Ekonomik

Roadster

Autorzy projektu: Jakub Frączek, Kamil Kras

Opiekun projektu: Wincenty Skwarek

Spis treści

Geneza projektu

Pewnie każdy młody chłopak od najmłodszych lat marzy o tym aby zasiąść za sterami samochodu wyścigowego, ciężarowego, statku latającego takiego jak śmigłowiec, samolot pasażerski bądź myśliwiec lub też może statku kosmicznego. Autorzy projektu mieli podobne marzenia i postanowili je spełnić budując symulator dzięki któremu nie tylko można mile spędzić wolny grając w ulubione gry, ale także przy okazji wiele się nauczyć.

Przeglądając zasoby Internetu można spotkać się z wieloma istniejącymi rozwiązaniami dotyczącymi budowy pojazdów jeżdżących czy latających. Większość konstrukcji współpracuje z istniejącymi grami komputerowymi wykorzystując do wyświetlania obrazu monitory ekranowe. Drugą grupę stanowią konstrukcje wyświetlające świat gry za pomocą gogli wirtualnej rzeczywistości czy obraz na ekranie za pomocą projektora. Wybór monitora do prezentacji „świata gry" w tym przypadku wydaje się być uzasadniony ze względu na możliwość skalowania rozwiązania w zależności od środków finansowych przeznaczonych na budowę konstrukcji.

Kolejną kwestią jest odzwierciedlenie realistyki pojazdu „design wnętrza" – w tym przypadku biorąc pod uwagę aspekt ekonomiczny najlepiej jest wybrać samochód osobowy ze względu na łatwą możliwość zdobycia części wyposażenia kabiny na rynku wtórnym bądź szrotach samochodowych.

Następnym problemem jest zastosowanie kontrolerów. Można tutaj skorzystać z gotowych fabrycznych kontrolerów jednak te dobrej jakości są zazwyczaj bardzo drogie.

Ostatnim do rozwiązania problemem, jeszcze przed przystąpieniem do budowy symulatora, jest kwestia jakiego rodzaju ma być to symulator. Wybór padł na samochód ciężarowy ze względu na to, że autorzy już niebawem będą posiadaczami prawa jazdy kategorii B więc będą prowadzić prawdziwe pojazdy natomiast na dzień dzisiejszy możliwość prowadzenia samochodu ciężarowego jest poza ich zasięgiem.

Reasumując decyzja padła na samochód ciężarowy współpracującego z grą Euro Truck Simulator 2 z wyposażeniem stanowiska kierowcy ze samochodu osobowego i wyświetlaniem obrazu na monitorze ekranowym. Uzyskany budżet na budowę symulatora to 1000 zł.

Założenia konstrukcyjne i ogólny opis projektu

Założenia konstrukcyjne

Z założenia projekt ma jak najwierniej, o ile to możliwe, odzwierciedlać oryginał łącznie z realistyką jazdy tzn. dźwięk pracy maszyny, odczucia związane z wciskaniem pedałów, ruchem kierownicy, włączania biegów, włączanie oprzyrządowania czy też doznawane przeciążenia podczas jazdy. W wyniku analizy istniejących rozwiązań i prowadzonych dyskusji w grupie wraz z opiekunem projektu ustalono poziomy funkcjonalności symulatora i etapy budowy.

Etap I (realizowany w ramach projektu do konkursu „Od pomysłu do przemysłu")

- pozyskanie podzespołów wyposażenia kabiny kierowcy: fotel, kokpit, kierownica, kolumna kierownicza, pedały sprzęgła hamulców i gazu, manetki świateł, kierunkowskazów i wycieraczek, zegary,
- budowa ramy „pojazdu" z kątownika przystosowanej do ewentualnej rozbudowy w kolejnych etapach rozwoju projektu,
- montaż pozyskanych podzespołów do ramy,
- montaż sensorów umożliwiających realistyczne sterowanie symulatorem,
- montaż podzespołów poprawiających realistkę gry: deska rozdzielcza z zegarami, układ generujący siłę reakcji kierownicy,
- montaż elektroniki,
- wykonanie i montaż tapicerki,
- programowanie układu sterowania i elektroniki układów wykonawczych,
- testy.

Etap 2 (Ewentualny rozwój projektu już po konkursie w zależności od pozyskanych środków finansowych):

- zakup trzech monitorów do obserwacji sytuacji na drodze i otoczenia samochodu wraz z mocniejszą kartą graficzną,
- budowa uchwytu monitorów,

Etap 3 (Dalszy rozwój projektu w przypadku znalezienia sponsorów):

- budowa platformy do zamocowania silników wraz z przekładniami ślimakowymi i sensoryką generującymi przyspieszenia w celu poprawy wrażeń gry,
- montaż silników z oprzyrządowaniem,
- zaimplementowanie układu sterowania w/w silnikami.

Pod pojęciem zakończenia prac nad projektem rozumie się zakończenie I etapu prac.

Ogólny opis projektu

Ekonomik Roadster - to nieruchomy symulator samochodu ciężarowego, który został stworzony z myślą o osobach, starających się uzyskać uprawnienia w kierowaniu samochodem, ale także kierowcach pragnących poprawić swoje umiejętności kierowania pojazdami. Projekt ma także pełnić rolę promocyjną Zespołu Szkół nr 4 w Jaśle.

Przystępując do etapu projektowania założono mechatroniczne podejście w projektowaniu tzn. wszystkie elementy konstrukcji były równolegle projektowane z wykorzystaniem oprogramowania CAD.

Metalowa rama została wykonana z kątownika hutniczego 35x35x4 w trzech częściach, dzięki czemu istnieje możliwość transportu symulatora przy wykorzystaniu samochodu osobowego typu kombi. Poszczególne elementy ramy zostały zespawane tworząc moduły skręcane za pomocą śrub M8x25. Do zmontowanej ramy został zamontowany fotel od samochodu VOLVO, pedały sprzęgła, hamulca i gazu wykonane z płaskownika 30x4 i blachy 3 mm. Następnie została zamontowana kolumna kierownicza od samochodu BMW E36 z modyfikacją umożliwiającą podpięcie mechanicznej części enkodera

inkrementalnego, większego koła przekładni łańcuchowej do sprzężenia zwrotnego kierownicy oraz mechanizmu łańcuchowego do ogranicznika kierownicy. Po zamontowaniu kolumny kierowniczej przykręcono do ramy silnik DC o mocy 200 W z małym kołem zębatym będącym w komplecie.

Kolejnym krokiem był montaż sensorów: potencjometry 10k obrotowe o charakterystyce liniowej do pedałów, enkodera inkrementalnego 600 imp./obr, manetek kierunkowskazów i wycieraczek od BMW E36 oraz zaprojektowanej i wykonanej skrzyni biegów – jej szczegółowa konstrukcja została opisana w dalszej części niniejszego dokumentu.

Po zamontowaniu sensoryki przykręcono elektronikę której główną część stanowią dwa mikrokontrolery arduino: arduino pro micro – pełniącego rolę sterownika sygnałów wejściowych oraz arduino nano – pełniącego rolę sterowania danych telemetrycznych gry. Opis części elektronicznej został przedstawiony w dalszej części niniejszej pracy.

Rys. 1 Symulator Ekonomik Roadster

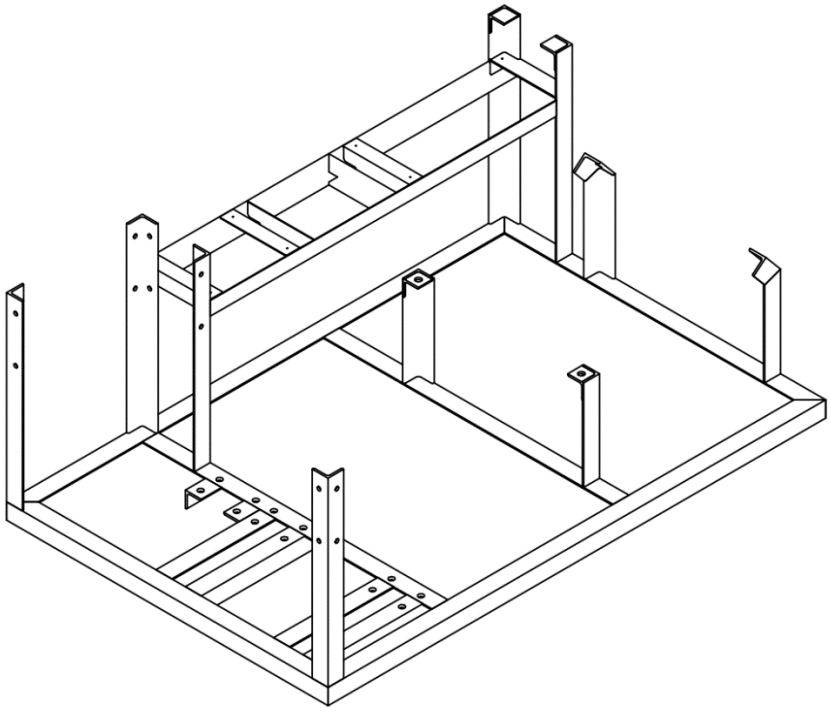
Arduino pro micro ma zaimplementowany autorski program sterujący napisany w języku C++ natomiast arduino nano korzysta z SimHuba (oprogramowania wspierającego takie projekty)

Całość została wykończona panelami ze sklejki o grubości 4 mm oklejonej ekoskórą w kolorze ciemnoszarym zbliżonym do oryginalnej okleiny kokpitu natomiast podłoga została wykonana ze sklejki o grubości 6mm i oklejona materiałem odpowiadającym materiałowi podłóg w seryjnych samochodach.

Część mechaniczna

Rama

Jak wspomniano wcześniej rama została wykonana z trzech części w celu łatwiejszego transportu symulatora. Dolna część ramy której projekt poglądowy przedstawiona na rys. 2 stanowi bazę do montażu fotela kierowcy, pedałów, skrzyni biegów i środkowej części ramy.

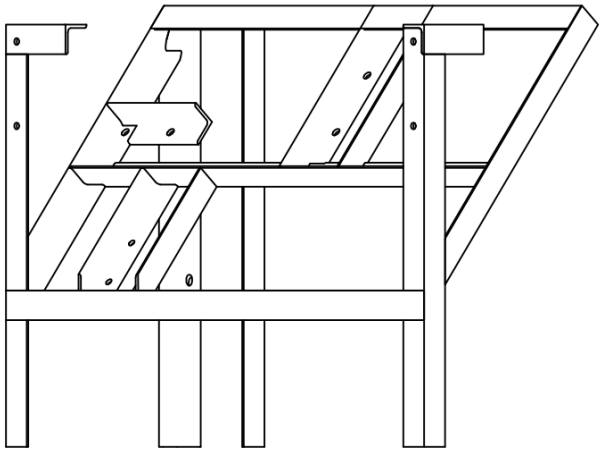


Rys. 2 Dolna część ramy rysunek poglądowy

Większość elementów tej części została wykonana z kątownika hutniczego 35x35x4 mm i pospawana. Wyjątek stanowi element znajdujący się po prawej stronie (górna część rys. 2) służący do montażu skrzyni biegów został on wykonany z kątownika zimnogiętego 30x30x2 mm. Rysunek wykonawczy dolnej

części ramy został dołączony do dokumentacji. Wykaz załączników znajduje się na końcu dokumentu.

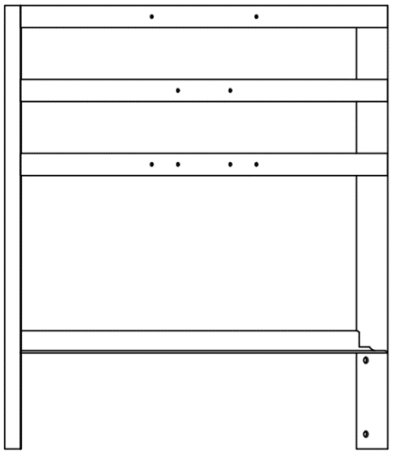
Środkowa część ramy przedstawiona na rys. 3 stanowi bazę do montażu kolumny kierowniczej, kokpitu, mechanizmu blokady krańcowej kierownicy, mechanizmu feedback kierownicy, enkodera, części elektronicznej oraz części górnej ramy.



Rys. 3 Środkowa część ramy rysunek poglądowy

Została ona wykonana z kątownika 35x35x4 mm z dodatkowym uchwytem na enkoder wykonanym z poszerzonej podkładki M16 rozwierconej do rozmiaru otworu wewnętrznego ϕ20.

Górna część ramy rys. 4 jest uchwytem umożliwiającym zamontowanie monitora o rozstawie śrub montażowych 100x100 i 200x200 mm w standardzie VESA. Ta część ramy wykonana została z kątownika giętego na zimno o rozmiarze 30x30x2 mm.



Rys. 4 Górna część ramy (uchwyt monitora) rysunek poglądowy

Wszystkie elementy po zespawaniu zostały dopasowane, a następnie oczyszczone i wymalowane na kolor czarny. Gotowe elementy zostały przedstawione na rys. 5a i rys. 5b

a) b)

Rys. 5 Pospawane i pomalowane elementy ramy

Pedały zostały wykonane z płaskownika 30x4 oraz blachy o grubości 3mm, a następnie pomalowane natryskowo na kolor czarny rys. 6.

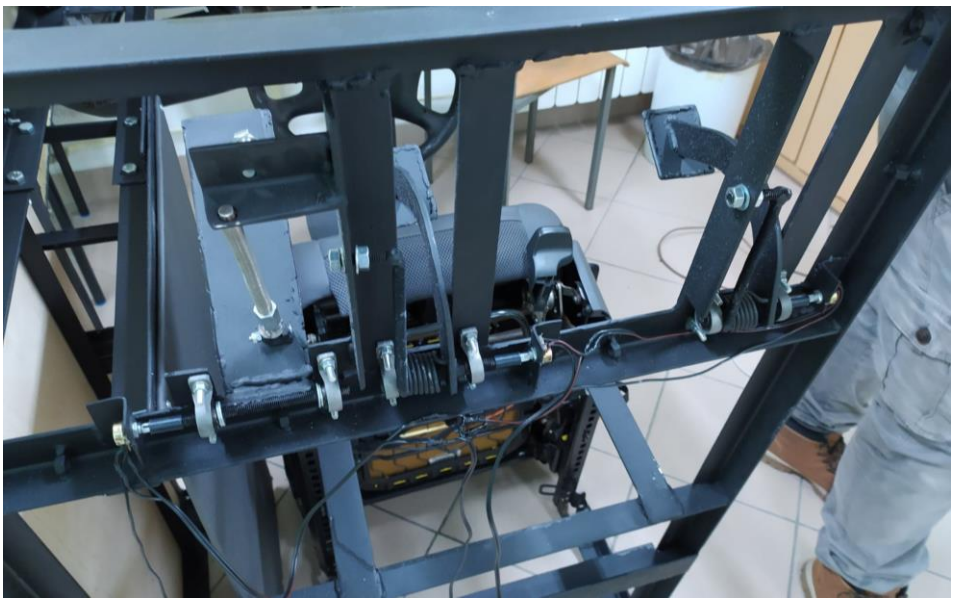
Rys. 6 Pospawane i pomalowane pedały

Montaż części mechanicznej

Pierwszą czynnością montażową było skręcenie dolnej i środkowej części ramy przy wykorzystaniu śrub stożkowych M8x25 rys. 7

Rys. 7 Proces skręcania dolnej i środkowej części ramy

W kolejnym kroku do ramy zostały zamontowane pedały przy użyciu łożysk samonastawnych KP08 w aluminiowej obudowie. Siłę reakcji stanowią sprężyny, które zostały wymontowane z uszkodzonej pralki automatycznej. Na rys. 8 zostały przedstawione zamontowane pedały wraz ze sprężynami i potencjometrami. Pomiędzy potencjometrem a osią pedału wstawiono gumową tuleję zapobiegającą powstawaniu naprężeń przy ewentualnej nieliniowości osi pedału z osią potencjometru.



Rys. 7 Finalny efekt montażu pedałów wraz z potencjometrami sterującymi.

Po zamontowaniu pedałów wykonano podłogę na wymiar ze sklejki o grubości 6 mm, a następnie oklejono ją materiałem imitującym materiał obiciowy podłóg samochodowych.

Do górnej części ramy przykręcono za pomocą łożysk samonastawnych w oprawie UCP 203 kolumnę kierowniczą od BMW E36. Rura aluminiowa kolumny kierownicy została skrócona, a do pozostałej części wmontowano pręt gwintowany M16 w celu łatwiejszego montażu kół zębatych ogranicznika ruchu kierownicy i zębatki sprzężenia zwrotnego kierownicy rys. 8.

Rys. 8 Zmodyfikowana kolumna kierownicza wraz z mechanizmem blokady w położeniach skrajnych.

Ogranicznik ruchu kierownicy w położeniach skrajnych został wykonany z łańcucha rowerowego i dwóch zębatek o średnicy 11mm. Podczas projektowania ramy założono że kierownica ma obracać się o kąt 900^0 co daje 2,5 obrotu. Jako element blokujący zamontowany na łańcuchu wykorzystano śrubę M4 wkręconą zamiast sworznia łączącego ogniwa. Śruba przy odpowiednim skrajnym położeniu blokuje się na ograniczniku. Wyznaczono odległość pomiędzy skrajnymi ogranicznikami:

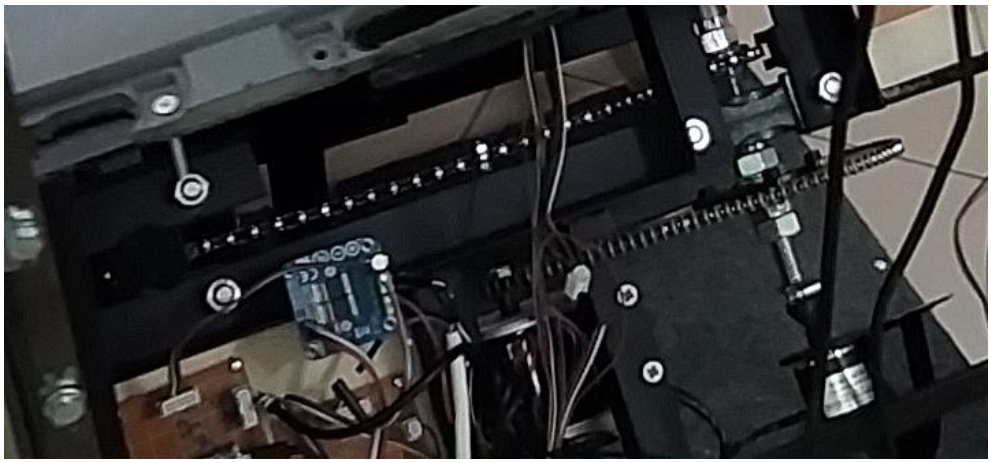
$$x = 2{,}5\pi d = 2{,}5 \cdot 3{,}14 \cdot 43mm = 337{,}55mm \approx 338mm$$

Ograniczniki zostały więc wspawane w odległości 338mm jeszcze na etapie spawania górnej części ramy.

Do generowania siły sprzężenia zwrotnego wykorzystano zestaw składający się z silnika DC o mocy 200W na napięcie 12V z zębatkami 11 i 80 zębów oraz łańcuchem 25H o długości 400 mm rys. 9.

Rys. 9 Mechanizm napędu sprzężenia zwrotnego kierownicy

Otrzymane przełożenie $i = \frac{80}{11} = 7{,}27$ pozwoliło zwiększyć moment siły działającej na wał kolumny kierowniczej korzystnie wpływając na pobór prądu, a przez to poprawiło wydolność układu zasilania sprzężenia zwrotnego. Zamontowany układ został przedstawiony na rys. 10.



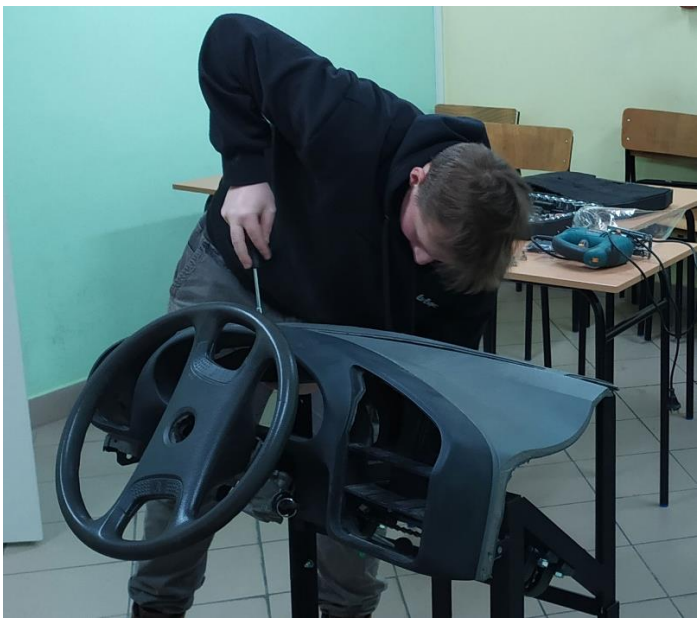
Rys. 10 Mechanizmy kierownicy: blokada skrajnych położeń, mechanizm sprzężenia zwrotnego, enkoder odczytu położenia.

Położenie kierownicy rejestrowane jest za pomocą enkodera inkrementalnego posiadającego 600 impulsów na obrót. Enkoder został przykręcony do ramy do specjalnie przygotowanego uchwytu i połączony z wałem za pomocą sprzęgła elastycznego wydrukowanego z gumy na drukarce 3D rys. 10.

Kolejną czynnością było docięcie na wymiar i zamontowanie kokpitu. Proces ten został przedstawiony na rys. 11 i rys. 12

Rys. 11 Kokpit przed docięciem na wymiar.

Na rys. 11 został przedstawiony wstępnie przycięty kokpit który został dopasowany do ramy i przycięty na wymiar. Na rys. 12 przedstawiono kokpit po docięciu.

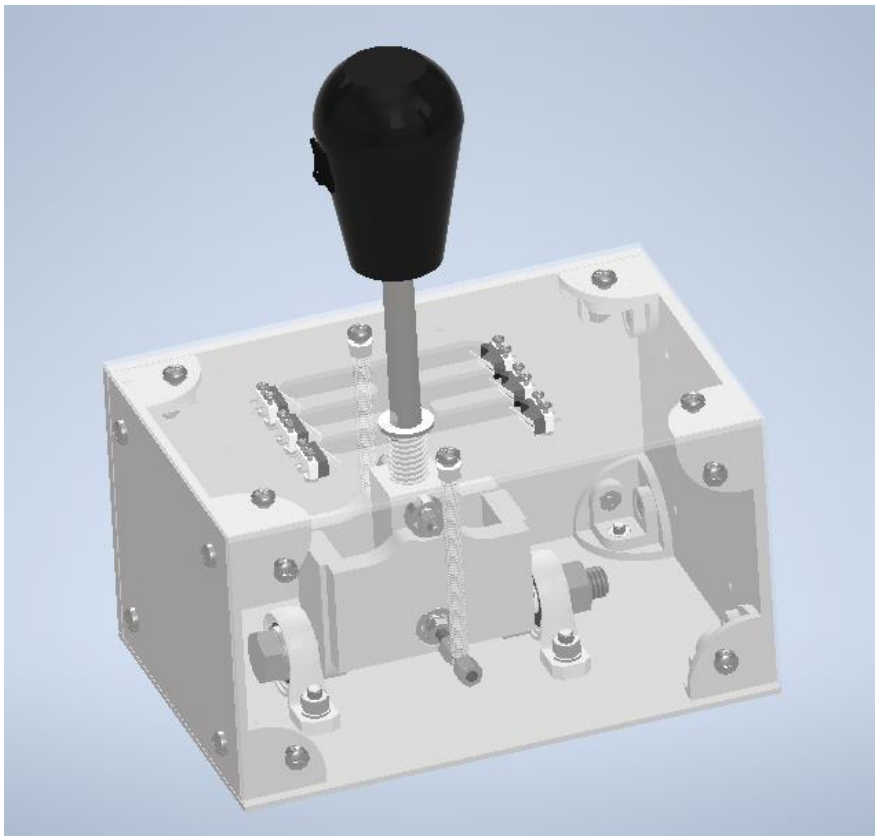


Rys. 12 Montaż dociętego kokpitu.

Skrzynia biegów

Z założenia symulator ma jak najwierniej odzwierciedlać prawdziwy samochód więc postanowiono zbudować skrzynię biegów manualną 6-cio biegową (5 biegów do przodu i 1 wsteczny) ponadto gałka lewarka jest wyposażona w przełącznik skrzyni górna-dolna tak jak w samochodzie ciężarowym.

Całość została zamodelowana w programie Inventor czego zaletą było dokładne dopasowanie części oraz utworzenie elementów, które następnie zostały wydrukowane na drukarce 3D będącej na wyposażeniu szkoły.



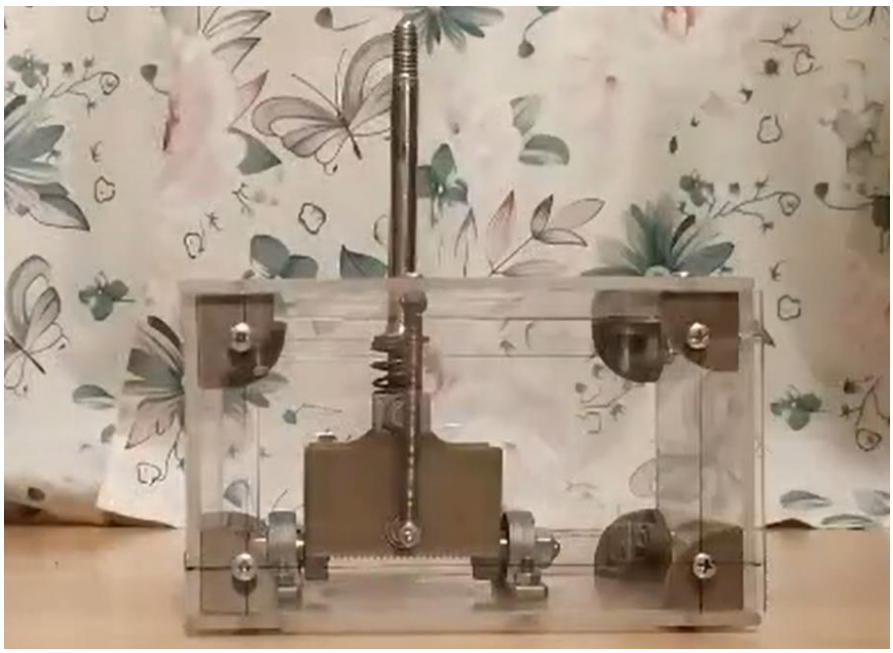
Rys. 13 Model skrzyni biegów wykonany w Inventorze.

Obudowa skrzyni została wykonana z przeźroczystej PLEXI o grubości 4mm. I skręcona za pomocą narożników wydrukowanych z PLA przy pomocy śrub M4x12. Do dolnej części obudowy zostały przykręcone łożyska samonastawne KP08 za pomocą śrub o łbach stożkowych M5x16. Mechanizm ruchu lewarka na boki zamocowany jest za pomocą śruby M10x110 i zblokowany z drugiej strony nakrętką. Lewarek utrzymywany jest w położeniu środkowym

dzięki dwom sprężynom o średnicy druta 1 mm i średnicy wewnętrznej 5 mm. Dwie sprężyny otrzymano po przecięciu jednej dłuższej i dospawaniu na odciętej części nakrętek M4, w które są wkręcane śruby mocujące M4x12. W dolnej części ucha sprężyn są zamontowane do sworznia wykonanego z pręta gwintowanego M4 który jednocześnie mocuje lewarek w mechanizmie. Pręt ten jest łożyskowany w celu poprawy płynności działania skrzyni oraz wydłużenia jego żywotności.

Mechanizm zatrzasku biegu składa się z dwóch części: dolnej i górnej. Część dolna mechanizmu jest tak wyprofilowana, aby łożyska zamontowane do części górnej przesuwając się blokowały bieg w odpowiednim położeniu. Efekt ten uzyskano dzięki sprężynie dociskowej 17,6x32 wykonanej z materiału o średnicy 1,5 mm. Górna część sprężyny blokowana jest za pomocą podkładki M10 przyspawanej do pręta lewarka o średnicy 10mm i gwintowanego w górnej części w celu umożliwienia montażu gałki lewarka. Gałka lewarka została wydrukowana na drukarce 3D z PLA i posiada otwór do montażu przełącznika skrzynia górna-dolna. Kanał do podciągnięcia przewodu do przełącznika oraz gwintowany otwór M10 do montażu na lewarku.

Do wczytania biegów użyto 6-ciu wyłączników krańcowych WK611 zamontowanych do górnej części obudowy za pomocą śrub M2x10 z nakrętkami.



Rys. 14 Zmontowany mechanizm skrzyni biegów.

Rysunek złożeniowy skrzyni i wszystkie rysunki wykonawcze zostały dołączone do dokumentacji jako załączniki i znajdują się w wykazie załączników.

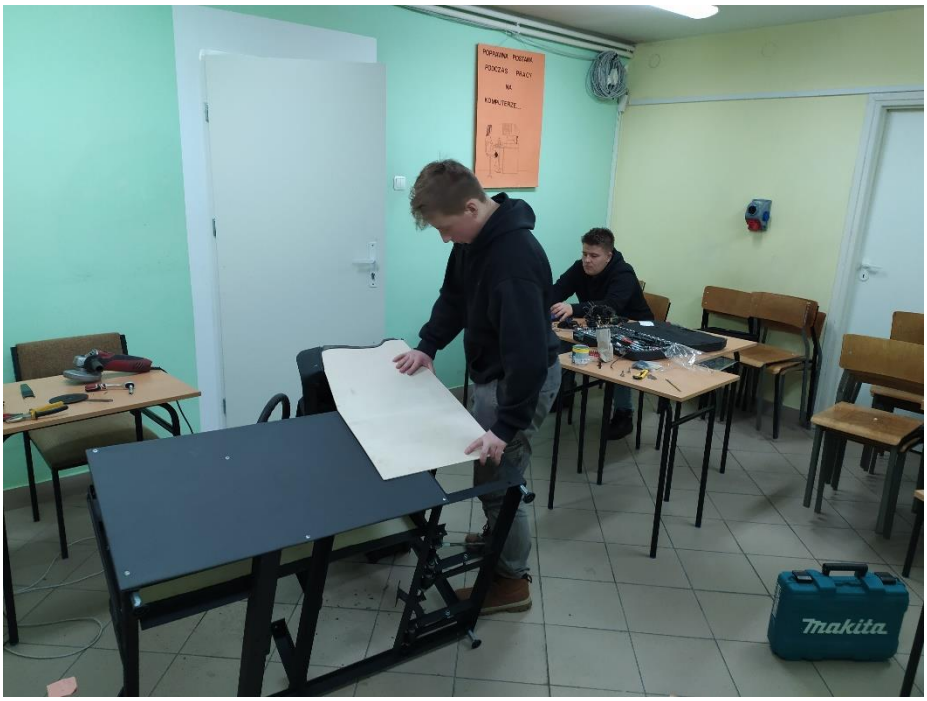
Tapicerka symulatora

Ostatnim etapem prac związanym z częścią mechaniczną było wykonanie paneli dostosowanych rozmiarem do rozmiaru ramy oklejenie ich skórą ekologiczną i przykręcenie do ramy. Panele zostały wykonane ze sklejki o grubości 4 mm. Materiał sklejka został wybrany ze względu na cenę i łatwość obróbki. Ekoskórę dobrano kolorystycznie do kokpitu samochodu.

Rys. 15 Docinanie ekoskóry do rozmiarów panelu.

Ze względu na brak modelu 3D kokpitu (szkoła niestety nie dysponuje skanerem 3D) i ewentualne zmiany kształtu ramy podczas spawania ramy (spawanie nie było wykonywane przez profesjonalnego spawacza) docinanie elementów tapicerki odbywało się na bieżąco rys 15 w celu uniknięcia ewentualnych niedokładności takich jak np. szczeliny związane z odkształceniem ramy podczas spawania. Materiał obiciowy został przyklejony do paneli przy użyciu kleju butapren, który jest dedykowany do klejenia tego typu materiałów. Po wyschnięciu panele zostały

przykręcone do ramy przy użyciu śrub M4x12 wyposażonych w łeb kulowy rys. 16.



Rys. 16 Montaż tapicerki obiciowej.

Panel za monitorem został wykonany z szarej płyty HIPS o grubości 2 mm na której za pomocą farby w sprayu została naniesiona nazwa ROADSTER rys. 17.

Rys. 17 Zmontowany projekt symulatora ROADSTER.

Podsumowując, otrzymano modułową konstrukcję mechaniczną pozwalającą na ewentualny transport do innych jednostek niż Zespół Szkół nr 4 w Jaśle za pomocą większego samochodu osobowego np. z nadwoziem typu kombi. Montaż całości przez 2 osoby po rozmontowaniu nie powinien zająć więcej niż 60 min dzięki czemu projekt będzie mógł być wystawiany w różnych pokazach i zdaniem autorów oraz opiekuna będzie stanowił bardzo dobry element promujący szkołę, a zwłaszcza liceum o profilu politechnicznym.

Część elektryczna

Analizując dostępne w Internecie rozwiązania zdecydowano się na zastosowanie kontrolerów Arduino ze względu na przystępną cenę klonów, które można zakupić za pośrednictwem serwisu Allegro. Zwrócono także uwagę na to aby całość konstrukcji, nie tylko mechanicznej, była modułowa. W związku z tą sytuacją podzielono część elektryczną na zadania jakie ma ona realizować:

- zasilanie podzespołów,
- interpretacja i przetwarzanie sygnałów wejściowych: pedały, skrzynia biegów, manetki, stacyjka itp.,
- interpretacja i przetwarzanie sygnałów wyjściowych z gry (telemetryka), prędkość, obroty silnika, stan paliwa, temperatura cieczy chłodzącej, obsługa lampek sygnalizacyjnych,
- sterowanie silnikiem sprzężenia zwrotnego.

Zaprojektowany schemat instalacji elektrycznej przy użyciu oprogramowania KiCAD został dołączony do dokumentacji i znajduje się w wykazie załączników.

Zasilanie

Symulator ze względu na zastosowanie mocnego silnika sprzężenia zwrotnego wymaga użycia zasilacza o dużej wydajności prądowej. Przeprowadzone testy pokazały zapotrzebowanie prądowe sięgające 5A przy aktywnym sprzężeniu zwrotnym przy zasilaniu silnika napięciem 12V ponadto

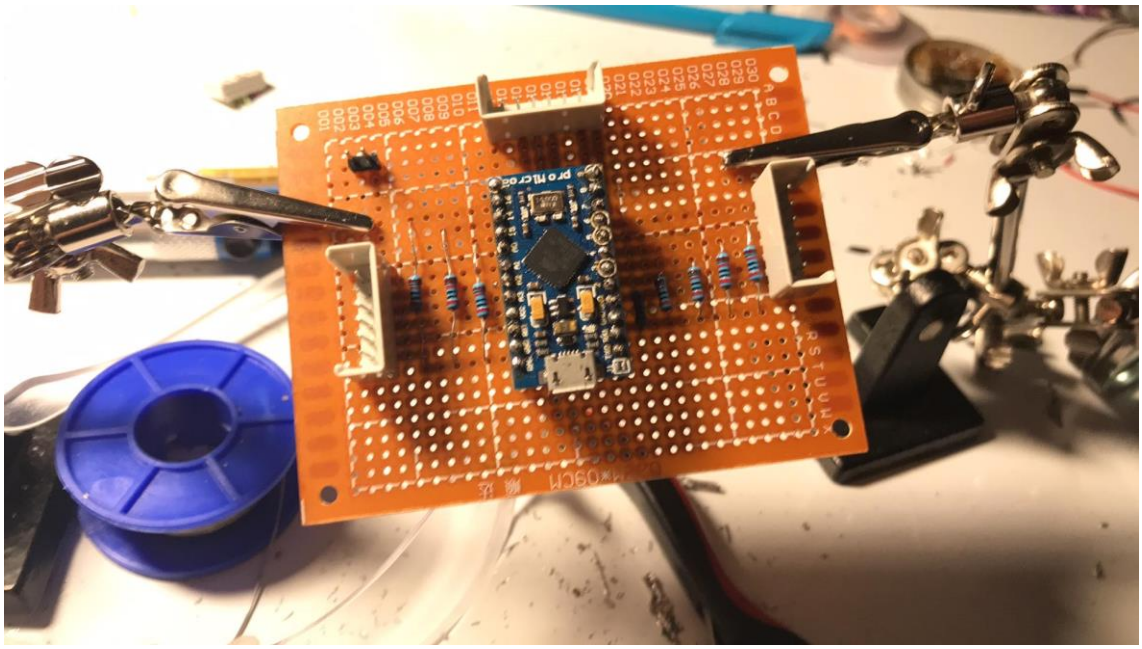
uwzględniając moc silnika 200W można wyznaczyć orientacyjny maksymalny prąd który wynosi.

$$I = \frac{P}{U} = \frac{200W}{12V} = 16{,}6A$$

Zapas takiej mocy można uzyskać stosując zasilacz komputerowy o mocy 700 W, który na linii +12V oferuje 19A prądu. Ponadto zasilacz posiada inne linie napięciowe. W przypadku realizowanego projektu wykorzystywana będzie jeszcze linia +5V celu zasilania logiki by nie obciążyć prądowo wejść USB.

Logika sygnałów wejściowych

Do sterowania sygnałami wejściowymi wykorzystano Arduino pro micro oparty na mikrokontrolerze Atmega32u4 16 z zegarem 16MHz dzięki czemu może być wykryte jak joystick. Podobnie działa Arduino Leonardo jednak zdecydowano się na pro micro gdyż można je obsadzić na płytce w zaprojektowanym gnieździe i w razie uszkodzenia szybko wymienić na inne rys. 18.



Rys. 18 Płytka sterująca wejściem sygnałów.

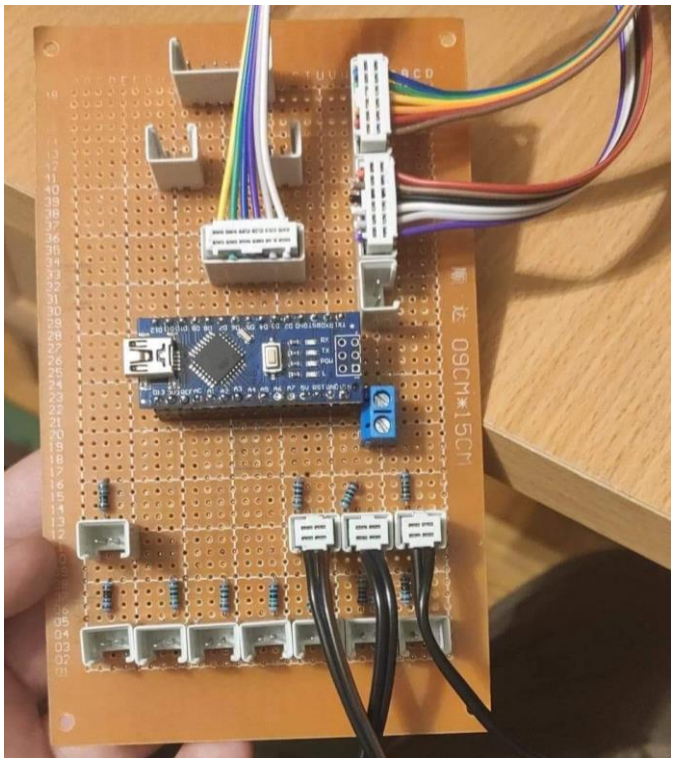
Na płytce przedstawionej na rys. 18 znajdują się trzy gniazda. Gniazdo z lewej strony płytki służy do podłączenia potencjometrów pedałów podłączonych do wejść A1, A2, A3 mikrokontrolera. Gniazdo w górnej części płytki wykorzystywane jest do podłączenia skrzyni biegów z której wyprowadzone jest

6 przewodów (5 sygnałowych i 1 zasilający) obsługujących programowo 10 stanów co zostało opisane w dalszej części niniejszego dokumentu. Mechanizm biegów wykorzystuje piny 10, 14, 16 natomiast przełącznik skrzyni D8, a hamulec ręczny D9. Gniazdo z prawej strony płytki wykorzystywane jest do podłączenia manetek oraz sterowania PWMem silnika DC, manetki używają pinów D2, D3, D4, D7 natomiast PWMy D5 i D6.

Wewnątrz skrzyni znajduje się płytka z układem diód 1N4007 co umożliwia uzyskanie 8 stanów za pomocą trzech pinów danych jest to autorskie rozwiązanie. Schemat tego rozwiązania przedstawiono na schemacie elektrycznym zamieszczonym w załączniku do dokumentu.

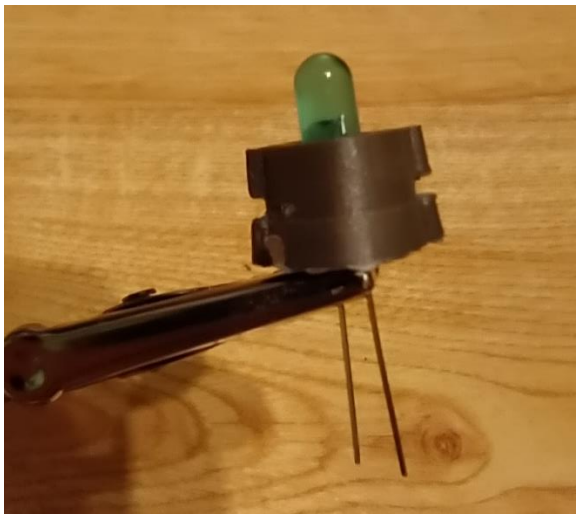
Logika sygnałów wyjściowych

Do sterowania sygnałami wyjściowymi użyto Arduino nano ze względu na możliwość zamontowania go w gnieździe i w razie uszkodzenia szybkiej wymiany podzespołu. Do osadzenia Arduino użyto płytki prototypowej nanosząc ścieżki i lutując gniazda rys. 19.



Rys. 19 Płytka sterująca wyjściem sygnałów.

Gniazdo ARK znajdujące się w centralnej części płytki wykorzystywane jest do podłączenia napięcia 12V zasilającego zegary. Górna część płytki wyposażona jest w trzy gniazda (aktualnie podłączone na zdjęciu): lewe sygnały sterowania wskazówkami zegarów, prawa strona górne gniazdo to masy GND natomiast poniżej sygnały +12V. W dolnej części płytki podwójne gniazda służą do podłączenia diód jako lampek kontrolnych. Ze względu na konieczność stosowania przekaźników mechanicznych lub tranzystorowych zastąpiono oryginalne żarówki diodami. W celu montażu diody LED w miejsce żarówki zaprojektowano specjalne oprawy, które następnie wydrukowano za pomocą drukarki 3D rys. 20. Rysunki wykonawcze opraw zostały dołączone jako załączniki do dokumentacji.



Rys. 20 Lampka sygnalizacyjna wykonana na bazie diody LED.

Sterowanie silnikiem sprzężenia zwrotnego

Do sterowania silnikiem wykorzystano mostek H BTS7960B o dużej wytrzymałości prądowej 43A mostek ten steruje silnikiem DC za pomocą PWM. Konfiguracja ustawiona jest w ten sposób my można sterować dowolnymi wartościami PWM w obydwu kierunkach co w przyszłości będzie wykorzystane przy dalszych pracach nad symulatorem w kolejnych etapach rozbudowy.

Oprogramowanie

Zgodnie z tym co zostało napisane wcześniej symulator ma dwa układy sterujące oparte na dwóch mikrokontrolerach Arduino. Sytuacja wymusza zastosowanie dwóch programów obsługujących sygnały wejściowe i sygnały wyjściowe.

Kod zaimplementowany w arduino pro micro (Sygnał wejściowy)

W projekcie wykorzystano bibliotekę Joystick.h , którą można pobrać ze strony https://github.com/YukMingLaw/ArduinoJoystickWithFFBLibrary oraz drugą bibliotekę Encoder.h dostępną pod adresem internetowym https://github.com/PaulStoffregen/Encoder/blob/master/Encoder.h). Kod jest autorski, nie był on inspirowany innym projektem/kopiowany. Program może być wgrany do arduino pro micro lub arduino leonardo.

Program sterujący został zapisany za pomocą języka C++.

Opis kodu programu

Włączanie/ Wyłączanie elementów

Na początku kodu znajdują się stałe zdefiniowane za pomocą instrukcji preprocesora, w celu łatwego włączania/wyłączania poszczególnych elementów(kierownica, skrzynia biegów itd.)

```
#define GAS_ENABLED 1
#define CLUTCH_ENABLED 1
#define BRAKE_ENABLED 1
#define STEERING_ENABLED 1
#define GEARBOX_ENABLED 1
#define GEARBOX_CHANGE_ENABLED 1
#define WIPERS_ENABLED 1
#define LIGHTS_ENABLED 1
#define INDICATORS_ENABLED 1
#define ENGINE_ENABLED 1
```

Rys. 21 Kod programu – definiowanie stałych.

Klasa Element

Klasa Element udostępnia możliwość włączania/wyłączania danego elementu za pomocą zmiennej enabled oraz metodę “setPin()”, przyjmująca za argumenty kolejno tablicę pinów oraz rozmiar tablicy. Dzięki niej można przypisać potrzebne piny.

```
class Element{
  public:
    bool enabled;
    short *pin;
    int aSize;

    void setPin(short a[], int b ){
      aSize=b;
      pin = new short[aSize];
      for(int i=0; i<aSize; i++)
        pin[i]=a[i];
    }
};
```

Rys. 22 Kod programu – klasa Element.

Klasa Ranged

Klasa Ranged umożliwia zapisanie zakresu w którym może działać dany element przy pomocy zmiennych “value_from” oraz “value_to”.

```
class Ranged{
  public:
    int value_from;
    int value_to;
};
```

Rys. 23 Kod programu – klasa Ranged.

Klasa Element_Ranged

“Element_Ranged”, dziedziczy po klasie “Element” oraz po klasie “Ranged”. Dodatkowo korzysta z funkcji “setInputPin()”, która odpowiada za ustawienie danej tablicy pinów na tryb “INPUT” (pinMode(pin, INPUT)). Za jej pomocą tworzone są obiekty dla: kierownicy oraz pedałów,

```
class Element_Ranged: public Element, public Ranged{
    public:
      Element_Ranged(bool a, short b[], int c, int d, int e){
        enabled=a;
        setPin(b,c);
        value_from=d;
        value_to=e;

        if(enabled)
          setInputPin(b,c);

      }
};
```

Rys. 24 Kod programu – klasa Element_Ranged.

Klasa Element_NRanged

"Element_NRanged", dziedziczy po klasie "Element". Dodatkowo korzysta z funkcji "setInputPin()", która odpowiada za ustawienie danej tablicy pinów na tryb "INPUT" (pinMode(pin, INPUT)). Za jej pomocą tworzone są obiekty dla: skrzyni biegów, przełącznika zmiany skrzyni biegów, sygnału dźwiękowego, manetek, hamulca ręcznego i silnika.

```
class Element_NRanged: public Element{
    public:
      Element_NRanged(bool a, short b[], int c){
        enabled=a;
        setPin(b,c);

        if(enabled)
          setInputPin(b,c);
      }
};
```

Rys. 25 Kod programu – klasa Element_NRanged.

Tworzenie obiektów klas Element_Ranged i Element_NRanged

Tablice pinów są przekazywane za pomocą tablic pomocniczych, które przy uruchomieniu programu w "void setup()" są usuwane.

```
short *clutchPin = new short[1]{A2};
Element_Ranged clutch(CLUTCH_ENABLED,clutchPin,1,250,600); //600

short *brakePin = new short[1]{A3};
Element_Ranged brake(BRAKE_ENABLED,brakePin,1,864,1001);

short *gasPin = new short[1]{A1};
Element_Ranged gas(GAS_ENABLED,gasPin,1,947,1023);

short *steeringPin = new short[2]{0,1};
Element_Ranged steering(STEERING_ENABLED ,steeringPin,2,-800,800);

short *gearboxPin = new short[3]{10,16,14};
Element_NRanged gearbox(GEARBOX_ENABLED,gearboxPin,3);

short *gearboxChangePin = new short[2]{8,9};
Element_NRanged gearboxChange(GEARBOX_CHANGE_ENABLED,gearboxChangePin,2);

short *indicatorsPin = new short[2]{2,3};
Element_NRanged indicators(INDICATORS_ENABLED,indicatorsPin,2);

short *lightsPin = new short[1]{4};
Element_NRanged lights(LIGHTS_ENABLED,lightsPin,1);

short *enginePiny = new short[2]{5,6};
Element_NRanged engine(ENGINE_ENABLED,enginePiny,2);

short *hornPin = new short[1]{7};
Element_NRanged horn(HORN_ENABLED,hornPin,1);

short *handbrakePin = new short[1]{9};
Element_NRanged handbrake(HANDBRAKE_ENABLED,handbrakePin,1);
```

Rys. 26 Kod programu – Tworzenie obiektów klas Element_Ranged i Element_Nranged.

Tworzenie obiektu joystick (bibliotek “joystick.h”)

Dzięki skorzystaniu z biblioteki “joystick.h”, arduino po podłączeniu do komputera zgłasza się jako joystick. Oś “Z” odpowiada za przekazywanie danych o położeniu kierownicy. Oś “Rx” odpowiada za przekazywanie danych o położeniu pedału hamulca. Oś “Rz” odpowiada za przekazywanie danych o położeniu pedału sprzęgła. Oś “Ry” odpowiada za przekazywanie danych o położeniu pedału gazu.

```
Joystick_ Joystick(JOYSTICK_DEFAULT_REPORT_ID,JOYSTICK_TYPE_JOYSTICK,28, 0,
  false, //x
  false, //y
  true, //z
  true, //rx
  true, //ry
  true, //rz.
  false, //rudder
  false, //throttle
  false, // accelerator
  false, // brake
  false // steering
  );
```

Rys. 27 Kod programu – Tworzenie obiektu joystick

"Joystick" posiada metodę "setButton()", dzięki której można przekazywać dane o stanie kierunkowskazów, skrzyni biegów itd.

Numer przycisku	Za co odpowiada
0	Bieg wsteczny
1	Bieg 1
2	Bieg 2
3	Bieg 3
4	Bieg 4
5	Bieg 5
6	Bieg 6
7	-
8	Przełączenie skrzyni biegów 1
9	Przełączenie skrzyni biegów 2
10	Kierunkowskaz lewy
11	Kierunkowskaz prawy
12	Światła
13	Wycieraczki

Funkcja setInputPin()

Ustawia każdy pin z otrzymanej tablicy na “INPUT”

```
void setInputPin(short pin[], int aSize){

  if(aSize>1){
    for(int i=0; i<aSize; i++)
      pinMode(pin[i],INPUT);
  }
  else
    pinMode(pin[0], INPUT);

}
```

Rys. 28 Kod programu – funkcja setInputPin()

Funkcja Gearbox()

Odczytuje wartości z trzech pinów, w celu zaoszczędzenia pinów zastosowano układ z diodami:

BIeg	Pin bA	Pin bB	Pin bC
0	0	0	0
1	0	0	1
2	1	0	1
3	0	1	1
4	1	0	0
5	0	1	0
6	1	1	1

Funkcja dodatkowo korzysta z zmiennych pomocniczych: “gear” (zawiera tablicę z numerami przycisków) oraz “lastGear” (w celu zapobiegnięcia wykonywania niepotrzebnych instrukcji w przypadku odczytywania tego samego biegu).

Funkcje Clutch(), Brake() oraz Gas()

Odpowiadają za odczytywanie wartości z potencjometrów, przymocowanych do pedałów i aktualizowanie odpowiednich osi.

```
void Gas(){
  rYValue=analogRead(gas.pin[0]);
  if(rYValue!=currentRyValue){
    Joystick.setRyAxis(rYValue);
    currentRyValue=rYValue;
  }
}
```

Rys. 29 Kod programu – funkcja Gas()

```
void Clutch(){
  rZValue=analogRead(clutch.pin[0]);
  if(rZValue!=currentRzValue){
    Joystick.setRzAxis(rZValue);
    currentRzValue=rZValue;
  }
}
```

Rys. 30 Kod programu – funkcja Clutch()

```
void Brake(){
  rXValue=analogRead(brake.pin[0]);
  if(rXValue!=currentRxValue){
    Joystick.setRxAxis(rXValue);
    currentRxValue=rXValue;
  }
}
```

Rys. 31 Kod programu – funkcja Brake()

Każda funkcja korzysta dodatkowo z dwóch zmiennych pomocniczych, dzięki którym ograniczone jest niepotrzebne wykonywanie instrukcji.

Kolejno “Gas()”, “Clutch()”, “Brake()”: “rYValue” i ”currentRyValue”, “rZValue” i ”currentRzValue”, “rXValue” i ”currentRxValue”.

Funkcja Steering()

Funkcja “Steering()” odpowiada za odczytywanie wartości z enkodera inkrementalnego przymocowanego do kierownicy, i aktualizację osi “Z”. Odczytywanie wartości realizowane jest przy pomocy biblioteki “Encoder.h”.

```
void Steering(){
  long long enc;
  enc = Enc.read()/3.64;
  if (enc != posEnc)
  {
    steeringValue=enc;
    Joystick.setZAxis(steeringValue);
    posEnc = enc;
  }
}
```

Rys. 32 Kod programu – funkcja Steering()

Funkcja korzysta ze zmiennych pomocniczych “posEnc” (przechowuje ostatnią wartość) oraz “steeringValue” (aktualna wartość).

Funkcja EngineFfb()

Odpowiada za sprzężenie zwrotne do kierownicy, to że kierownica po puszczeniu wraca do pierwotnej pozycji. Na podstawie wartości “steeringValue”, czyli aktualnego położenia kierownicy, silnik dąży do skorygowania położenia kierownicy do wartości ok. 0, czyli pierwotnej pozycji. Sam silnik sterowany jest za pomocą pinów PWM.

```
void EngineFfb(){

  if(steeringValue>=-5&&steeringValue<=5&&engineLastPos==true){
    analogWrite(engine.pin[0],0);
    analogWrite(engine.pin[1],0);
    engineLastPos=false;
  }
  else if(steeringValue>5&&engineLastPos==false){
    analogWrite(engine.pin[1],50);
    engineLastPos=true;

    }
  else if(steeringValue<-5&&engineLastPos==false){
    analogWrite(engine.pin[0],50);
    engineLastPos==true;

  }
}
```

Rys. 33 Kod programu – funkcja EngineFfb()

Funkcja korzysta ze zmiennej pomocniczej “engineLastPos” (dzięki niej wartości na PWM podane są tylko raz)

Funkcja GearboxChange()

Odpowiada za zmianę trybu skrzyni biegów. Dane odczytywane są z przełącznika znajdującego się na skrzyni biegów.

```
void GearboxChange(){

  currentgearboxChange=digitalRead(8);
  if(currentgearboxChange==1&&lastgearboxChange==0){
      Joystick.setButton(8,0);
      Joystick.setButton(9,1);
      lastgearboxChange=currentgearboxChange;
  }
  else if(currentgearboxChange==0&&lastgearboxChange==1){
      Joystick.setButton(8,1);
      Joystick.setButton(9,0);
      lastgearboxChange=currentgearboxChange;
  }
}
```

Rys. 34 Kod programu – funkcja GearboxChange()

Funkcja korzysta z dwóch zmiennych pomocniczych “currentgearboxChange” (przechowuje aktualną pozycję przełącznika) oraz “lastgearboxChange” (ostatnia pozycja przełącznika).

Funkcje LeftIndicator(), RightIndicator(), Lights() i Wiper()

Odpowiadają kolejno za obsługę: prawego kierunkowskazu, lewego kierunkowskazu, przełącznika świateł oraz wycieraczek.

Funkcje działają na zasadzie odczytywania wartości 0/1. I pokazywania ich jako przyciski joysticka.

void setup()

Rozpoczęcie pracy joysticka. Ustawienie zakresów obiektów klasy “Element_Ranged”.

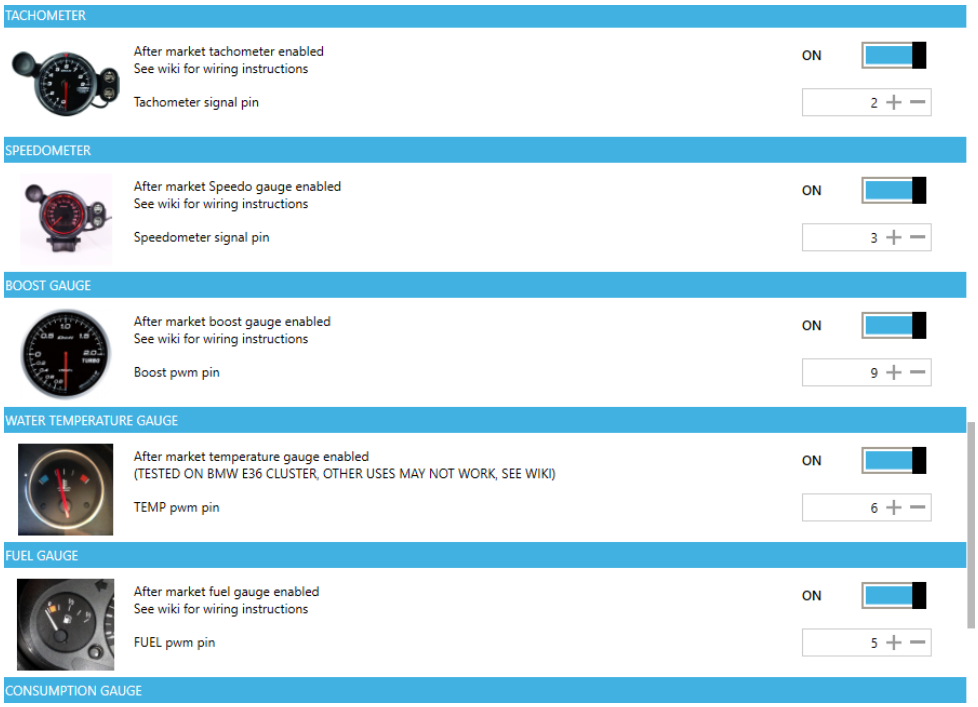
Zwolnienie pamięci zajmowanej przez tabele pomocnicze do pinów.

void loop()

Wywoływanie funkcji odpowiedzialnych za wszystkie elementy.

Kod zaimplementowany w Arduino nano (sygnał wyjściowy)

Ze względu na ograniczenia czasowe konkursu, dosyć krótki termin jak na tego rodzaju projekt, zdecydowano się użyć programu Sim Hub do generowania kodu sterującego sygnałami wyjściowymi (telemetrią gry). Oprogramowanie Sim Hub dostępne jest pod adresem https://www.simhubdash.com/ i jest bardzo wygodne dla osób nie znających języka C++. Interfejs oprogramowania jest w trybie graficznym co czyni pracę z programem bardzo przyjemną.



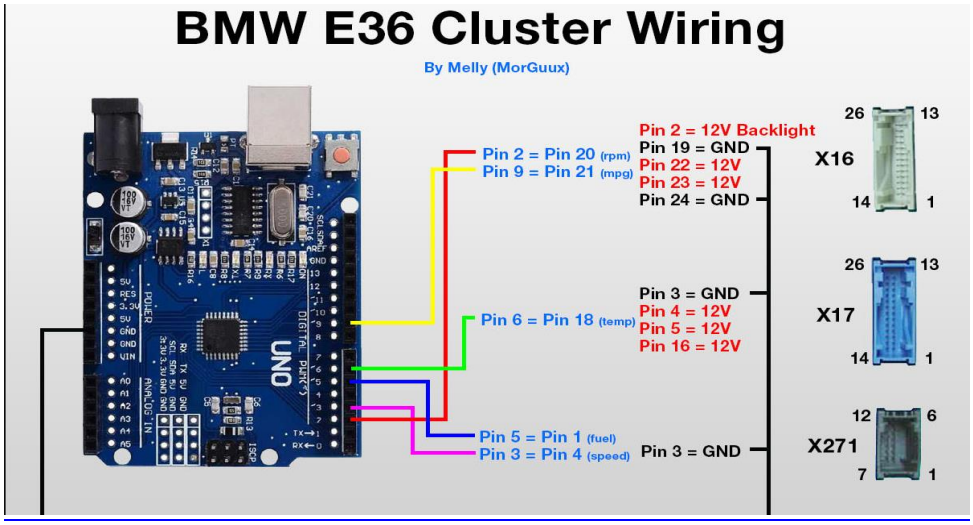
Rys. 35 Zrzut ekranu programu Sim Hub

Na rys. 35 przedstawiono interfejs programu Sim Hub wraz z oknem konfiguracji odpowiednich pinów arduino do których zostanie przypisana realizacja poszczególnych zadań.

Najprościej mówiąc Sim Hub przechwytuje dane telemetryczne z gry przesyła je na inne urządzenia w tym przypadku jest to Arduino.

W pierwszej kolejności zdefiniowano piny w aplikacji Sim Hub według schematu połączeń zegarów BMW E36. Skorzystano w tym przypadku z filmu

znajdującego się na stronie https://www.youtube.com/watch?v=EyVwf2w18Vs

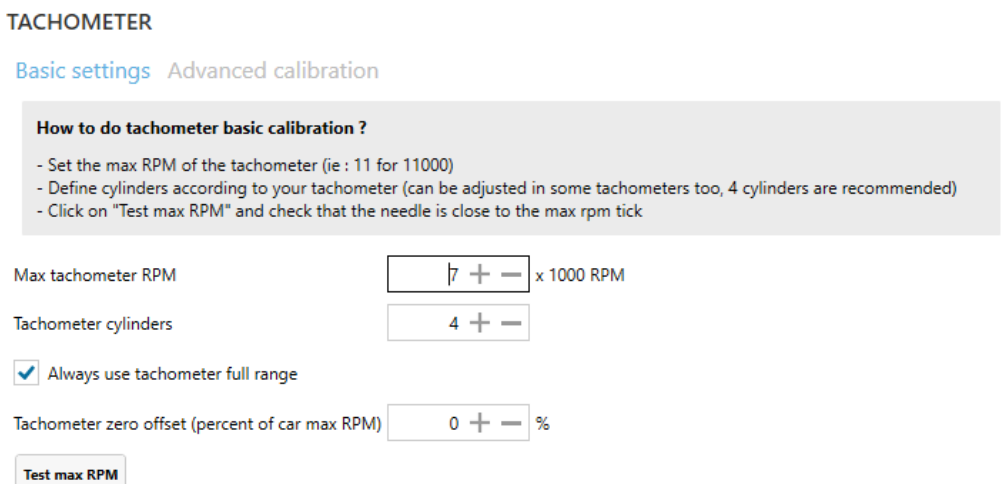


Rys. 36 Schemat podłączenia zegarów do mikrokontrolera arduino uno

W przypadku realizowanego projektu skorzystano zamiast z Arduino Uno z Arduino Nano ale obydwa są zbudowane w oparciu o mikrokontroler Atmega328 co czyni, że obydwa urządzenia są bardzo podobne.

Po przypisaniu pinów zgodnie z rys. 36 i schematem instalacji elektrycznej wgrano program do pamięci Arduino.

Kolejna czynność to ustawienie zakresów wartości wskaźników tak aby nie wychodziły poza skalę



Rys. 37 Ustawienie zakresu wskazywania obrotomierza

Dla przykładu na rys. 37 przedstawiono proces kalibracji obrotomierza mający na celu wyświetlanie wartości zgodnej z wartością odczytaną z gry. W podobny sposób skalibrowano pozostałe wskaźniki: prędkościomierza, wskaźnik paliwa i wskaźnik temperatury.

Następną czynnością było ustawienie pinów diód jako kontrolek w tym celu za pomocą Arduino IDE został wgrany do Arduino Nano plik custom protocol do obsługi diód LED.

Kod programu

W pierwszych liniach kodu zostały zdefiniowane zmienne w których będą przechowywane nazwy pinów arduino rys. 38.

```
int lewy;
int prawy;
int drogowe;
int mijania;
int reczny;

int lewyPin = A4;
int prawyPin = A1;
int drogowePin = A2;
int mijaniaPin = A0;
int recznyPin = A3;
```

Rys. 38 Kod programu – definicja pinów Arduino

Następnie ustawiono te piny jako wyjścia rys. 39

```
void setup()
{

pinMode(lewyPin,OUTPUT);
pinMode(prawyPin,OUTPUT);
pinMode(drogowePin,OUTPUT);
pinMode(mijaniaPin,OUTPUT);
pinMode(recznyPin,OUTPUT);

digitalWrite(lewyPin,LOW);
digitalWrite(prawyPin,LOW);
digitalWrite(drogowePin,LOW);
digitalWrite(mijaniaPin,LOW);
digitalWrite(recznyPin,LOW);

delay(2000);

digitalWrite(lewyPin,HIGH);
digitalWrite(prawyPin,HIGH);
digitalWrite(drogowePin,HIGH);
digitalWrite(mijaniaPin,HIGH);
digitalWrite(recznyPin,HIGH);

}
```

Rys. 39 Kod programu – definicja pinów Arduino jako wyjścia

Ostatnim elementem kodu było zapisanie odpowiednich zależności realizujących poszczególne zadania rys. 40

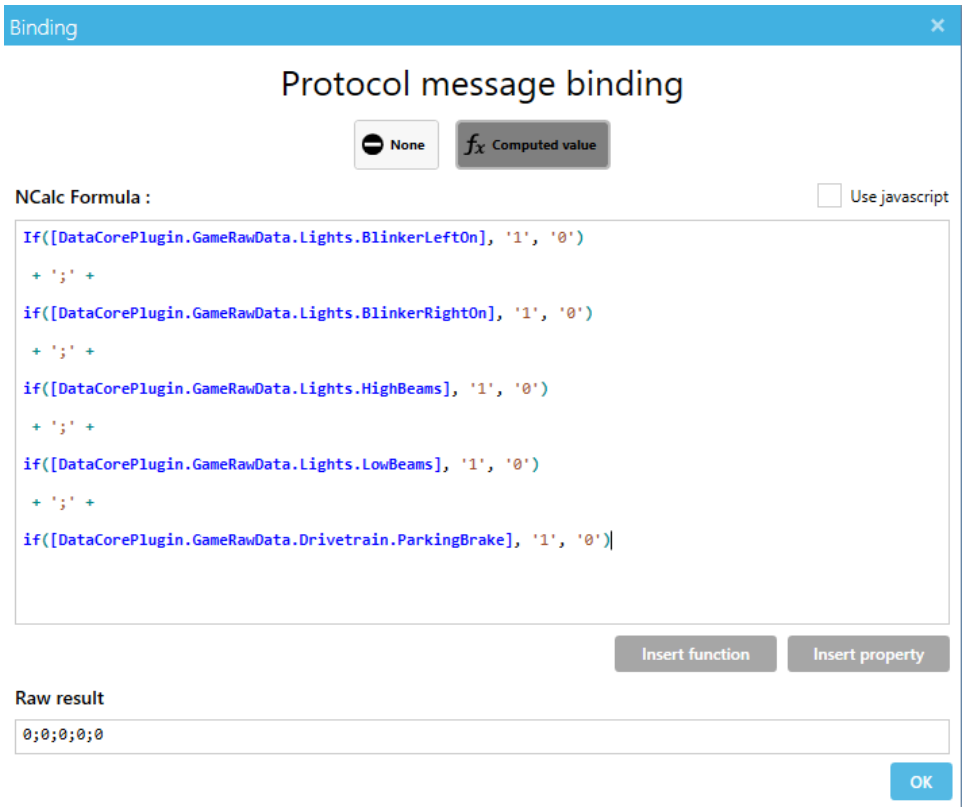
```
if(lewy == 1 && prawy == 0){
  digitalWrite(lewyPin,LOW);
  digitalWrite(prawyPin,HIGH);
}

else if (lewy == 0 && prawy == 1){
  digitalWrite(lewyPin,HIGH);
  digitalWrite(prawyPin,LOW);
}
else if (lewy == 1 && prawy == 1){
  digitalWrite(lewyPin,LOW);
  digitalWrite(prawyPin,LOW);
}

else{
  digitalWrite(prawyPin,HIGH);
  digitalWrite(lewyPin,HIGH);
}

if(drogowe == 1){digitalWrite(drogowePin,LOW);}

else{digitalWrite(drogowePin,HIGH);}

if(mijania == 1){digitalWrite(mijaniaPin,LOW);}

else{digitalWrite(mijaniaPin,HIGH);}

if(reczny == 1){digitalWrite(recznyPin,LOW);}

else{digitalWrite(recznyPin,HIGH);}
```

Rys. 40 Kod programu – część wykonywalna kodu

```
int lewy = FlowSerialReadStringUntil(';').toInt();
int prawy = FlowSerialReadStringUntil(';').toInt();
int drogowe = FlowSerialReadStringUntil(';').toInt();
int mijania = FlowSerialReadStringUntil(';').toInt();
int reczny = FlowSerialReadStringUntil('\n').toInt();
```

Rys. 41 Kod programu – część wykonywalna kodu

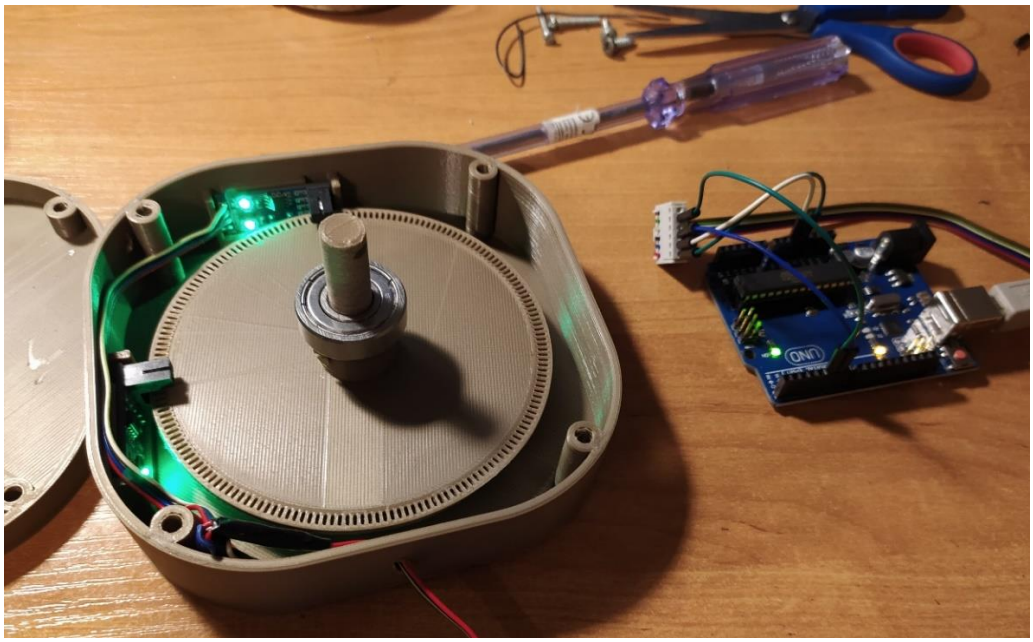
Kod który został przedstawiony na rys 38 – rys 41 został wklejony do Sim Huba w zakładce my hardware - Custom protocol rys. 42.



Rys. 42Zrzut ekranu programu SmHub po wgraniu kodu.

Testy

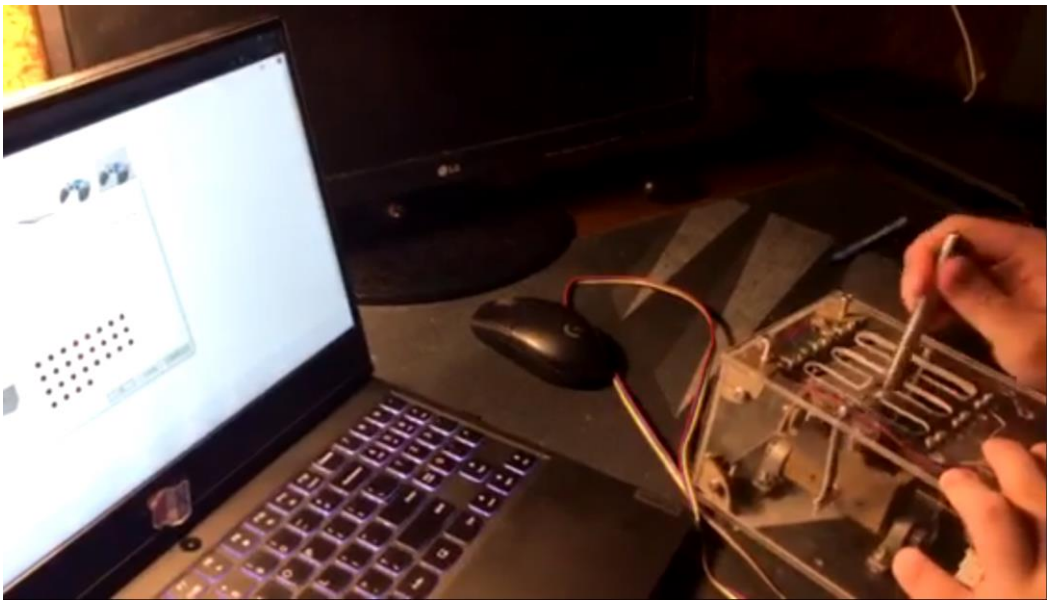
Czas budowy symulatora przypadł na okres kiedy szkoły były zamknięte w związku z tym podzielono pracę w ten sposób aby jeden z członków zespołu zajął się obsługą sygnałów wejściowych, a drugi sygnałów wyjściowych. Pierwsze testy dotyczyły możliwości użycia enkodera jako kontrolera położenia kierownicy i potencjometrów obrotowych o charakterystyce liniowej jako kontrolerów pedałów. Potencjometry sprawdziły się bardzo dobrze natomiast enkoder o 20 imp./obr miał zdecydowanie zbyt niskie parametry by go użyć w symulatorze. Zaprojektowano enkoder o 180 szczelinach i wydrukowano go na drukarce 3D rys. 43. Do budowy wykorzystano dwa moduły optyczne Arduino jeden jako kanał A, a drugi jako kanał B z odpowiednim ustawieniem aby sczytywały impulsy oraz kierunek ruchu rys. 43. Mimo że enkoder działał prawidłowo jednak swoimi wymiarami utrudniał montaż w ramie symulatora. Zdecydowano się na zakup enkodera fabrycznego z 600 imp./obr.



Rys. 43 Enkoder wydrukowany na drukarce 3D 180 imp/obr

Pierwsze próby biurkowe dotyczyły zachowania się gry na obrót enkodera czy potencjometrów oraz reakcji zegarów wyciągniętych z prawdziwego samochodu po podłączeniu do układu sterującego. Wszystkie testy przebiegły pomyślnie co było impulsem do dalszej pracy nad skrzynią biegów i konstrukcją ramy.

Rys. 44 Test zegarów od BMW E36



Rys. 45 Test skrzyni biegów

Po zmontowaniu ramy przeprowadzono szereg prób mających na celu prawidłowe skalibrowanie potencjometrów, prawidłowe działanie skrzyni biegów czy wskazań zegarów rys. 45 i rys. 46.

Rys. 46 Kalibracja potencjometrów pedałów

Kolejnym zadaniem było ustawienie odpowiedniej siły reakcji kierownicy w sprzężeniu zwrotnym poprzez dobranie odpowiednich wartości PWM. Obecnie działa on w trybie spring tzn. kierownica wraca do położenia centralnego.

Ostatnim etapem a zarazem najprzyjemniejszym były testy symulatora w grze. Ustawienie odpowiednich osi i „klawiszy” a także ich ostateczna kalibracja.

Praktyczne zastosowanie projektu

Od samego początku postawiono sobie za cel aby symulator mógł być wykorzystany nie tylko do rozrywki. Jego modułowa budowa pozwala na rozbudowę urządzenia w przyszłości. Już obecnie Ekonomik Roadster ma wiele zastosowań wśród których do najważniejszych można zaliczyć:

- rozrywka (możliwość zagrania po lekcjach),
- nauka poprzez zabawę – opracowywanie nowych rozwiązań konstrukcyjnych i algorytmów obsługujących podzespoły,
- ćwiczenie umiejętności jazdy,
- utworzenie społeczności szkolnej lub internetowej skupionej wokół projektu,
- promocja klasy liceum o profilu politechnicznym.

Podsumowanie

Nie każda szkoła może pochwalić się symulatorem w swoich murach. Naszej udało zbudować się taki. Pragniemy serdecznie podziękować osobom i instytucjom, które wsparły nas finansowo i rzeczowo podczas realizacji projektu Ekonomik Roadster

- Stowarzyszenie Oświatowo Edukacyjne Zespołu Szkół nr 4 w Jaśle
- Krzysztof Jędrzejczyk
- Adrian Błasik z firmy Autoszyk w Jaśle
- Tomasz Wojdyła z firmy Żelbud Sp z o.o.

Faza pierwsza projektu została zakończona jednakże naszym planem jest dalszy jego rozwój poprzez instalację większego wyświetlacza, dołożenie bocznych monitorów, zbudowanie platformy imitującej przeciążenia (siły bezwładności), napisanie własnego softu pobierającego dane z gry. Jak mówi nasz

opiekun „Jeśli będą chętni to jeszcze kilka pokoleń uczniów może pracować nad Roadsterem”.

Kosztorys

Nazwa elementu	jednostka	ilość	cena	Wartość
Arduino PRO MICRO ATmega32U4 AVR Leonardo 16MHz	szt.	1	24,00 zł	24,00 zł
Moduł Arduino NANO V3 ATmega328P V3.0 CH340	szt.	1	14,99 zł	14,99 zł
Cyna 1mm 100g Sn60Pb40	szt.	1	18,00 zł	18,00 zł
Enkoder inkrementalny NPN 600 imp/obr	szt.	1	79,99 zł	79,99 zł
Sterownik silnika BTS7960B DC 43A MOSTEK H	szt.	1	45,00 zł	45,00 zł
Wyłączników krańcowy WK611	szt.	6	1,90 zł	11,40 zł
Listwa goldpin żeńska 2.54mm 1x40 pin	szt.	2	2,00 zł	4,00 zł
Listwa goldpin męska 2.54mm 1x40 pin	szt.	2	2,00 zł	4,00 zł
Płytka Uniwersalna PCB 90x150 mm 9x15 cm	szt.	1	3,90 zł	3,90 zł
Płytka uniwersalna PCB 7x9 cm	szt.	3	2,30 zł	6,90 zł
Dioda 1N4007	szt.	7	1,20 zł	8,40 zł
Zasilacz komputerowy 700 W	szt.	1	80,00 zł	80,00 zł
MADRYT Eko Skóra Skaj Derma tapicerka meblowa	m2	2	14,30 zł	28,60 zł
Wtyk 6-pinowy męski	szt.	10	0,65 zł	6,50 zł
Wtyk 6-pinowy żeński	szt.	10	0,86 zł	8,60 zł
Wtyk 2-pinowy męski	szt.	18	0,41 zł	7,38 zł
Wtyk 2-pinowy żeński	szt.	18	0,53 zł	9,54 zł
Gniazdo ARK	szt.	2	1,29 zł	2,58 zł
Przewód micro USB	szt.	1	8,99 zł	8,99 zł
Przewód mini USB	szt.	1	9,99 zł	9,99 zł
Kabel wstążkowy 6x0,22 mm2	m	5	5,61 zł	28,05 zł
Klej Butapren	szt.	0,7	50,00 zł	35,00 zł
Sklejka 4mm	m2	2,5	25,00 zł	62,50 zł
Sklejka 6mm	m2	1,2	34,00 zł	40,80 zł
Silnik DC 12V 200W z przekładnią łańcuchową	szt.	1	130,00 zł	130,00 zł

Potencjometr obrotowy 10K Ohm B	szt.	3	1,65 zł	4,95 zł
Kątownik hutniczy 35x35x4	m	18	9,93 zł	178,74 zł
kątownik zimnogięty 30x30x2	m	6	7,20 zł	43,20 zł
Płaskownik 30x4	m	0,5	6,50 zł	3,25 zł
Blacha 3mm	m2	0,01	22,00 zł	0,22 zł
Sprężyna o średnicy druta 0,8 mm i średnicy wewnętrznej 5 mm	szt.	2	1,20 zł	2,40 zł
Farba w sprayu	szt.	1	9,50 zł	9,50 zł
Łożysko samonastawne KP08	szt.	5	8,00 zł	40,00 zł
UCP 203 Łożysko Samonastawne	szt.	2	12,37 zł	24,74 zł
Zębatka 11mm	szt.	2	5,00 zł	10,00 zł
Pręt gwintowany M16	szt.	1	9,66 zł	9,66 zł
Śruba M4x12 z łbem kulowym	szt.	30	0,10 zł	3,00 zł
Śruba M4x12	szt.	20	0,10 zł	2,00 zł
Śruba M5x16	szt.	4	0,15 zł	0,60 zł
Śruba M10x110	szt.	1	2,40 zł	2,40 zł
Śruba M8x25 z łbem stożkowym	szt.	8	0,12 zł	0,96 zł
Pręt gwintowany M4	szt.	1	1,75 zł	1,75 zł
Śruba M2x10	szt.	20	0,10 zł	2,00 zł
Śruba M4x12 wyposażonych w łeb kulowy	szt.	30	0,10 zł	3,00 zł
Nakrętka, podkładka M2	szt.	20	0,05 zł	1,00 zł
Nakrętka, podkładka M4	szt.	30	0,03 zł	0,90 zł
Nakrętka, podkładka M8	szt.	60	0,02 zł	1,20 zł
Nakrętka, podkładka M10	szt.	10	0,03 zł	0,30 zł
Nakrętka, podkładka M16	szt.	6	0,50 zł	3,00 zł
Sprężyna dociskowa 17,6x32 wykonanej z materiału o średnicy 1,5 mm	szt.	1	2,00 zł	2,00 zł
PLEXI 4mm	m2	0,05	159,00 zł	7,95 zł
Płyta HIPS 2 mm	m2	0,7	28,00 zł	19,60 zł
Wykładzina podłogowa	m2	0,7	20,00 zł	14,00 zł
Opornik 20k ohm	szt.	8	0,02 zł	0,16 zł
Opornik 4k7 ohm	szt.	2	0,02 zł	0,04 zł
Opornik 100 ohm	szt.	4	0,02 zł	0,08 zł
Opornik 200 ohm	szt.	3	0,02 zł	0,06 zł

Opornik 68 ohm	szt.	3	0,02 zł	0,06 zł
Dioda LED czerwona	szt.	3	0,30 zł	0,90 zł
Dioda LED zielona	szt.	2	0,30 zł	0,60 zł
Dioda LED żółta	szt.	2	0,30 zł	0,60 zł
Dioda LED niebieska	szt.	1	0,30 zł	0,30 zł
Dioda LED biała	szt.	3	0,30 zł	0,90 zł
Przełącznik dwupozycyjny	szt.	1	1,40 zł	1,40 zł
TAŚMA IZOLACYJNA CZARNA 12MM X 10M	szt.	1	2,49 zł	2,49 zł
Rurka termokurczliwa o średnicy 12mm i długości 1m	szt.	1	2,00 zł	2,00 zł
Filament PLA	szt.	1	40,00 zł	40,00 zł
Całkowity koszt:				**1 121,02 zł**

W kosztorysie nie uwzględniono elementów otrzymanych gratis takich jak: kokpit, fotel, kolumnę kierowniczą, kierownicę zegary, elementy ścierne, farba w puszce do malowania ramy, łańcuch rowerowy.

Wykaz rysunków i schematów :

Część mechaniczna

1. Skrzynia biegów złożenie złożenie.pdf
2. Górna pokrywa skrzyni biegów rys_1.pdf
3. Dolna pokrywa skrzyni biegów rys_2.pdf
4. Tylna pokrywa skrzyni biegów rys_3.pdf
5. Przednia pokrywa skrzyni biegów rys_4.pdf
6. Boczna pokrywa skrzyni biegów rys_5.pdf
7. Gałka lewarka skrzyni biegów rys_6.pdf
8. Mocowanie lewarka skrzyni rys_7.pdf
9. Drążek lewarka rys_8.pdf
10. Mocowanie łożysk drążka rys_9.pdf
11. Sworzeń montażu drążka lewarka rys_10.pdf
12. Narożnik do skręcenia skrzyni rys_11.pdf
13. Dolna część ramy symulatora rys_12.pdf
14. Środkowa część ramy symulatora rys_13.pdf

15. Górna część ramy symulatora rys_14.pdf
16. Oprawka diody mała rys_15.pdf
17. Oprawka diody duża rys_16.pdf

Część elektryczna

18. Schemat elektroniczny schemat_elektryczny.pdf

Kody programów

19. Kod arduino pro micro Ekonomik-Roadster.ino
20. Sim Hub - Custom protocol custom_protocol.ino