AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA AUTOMATYKI I INŻYNIERII BIOMEDYCZNEJ

Praca dyplomowa inżynierska

Wyłącznik awaryjny pojazdu autonomicznego
Emergency Stop Autonomous Vehicle

Autor: *Piotr Banaszkiewicz*
Kierunek studiów: *Automatyka i Robotyka*
Opiekun pracy: *dr inż. Marek Długosz*

Kraków, 2016

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo oszkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego", oświadczam, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i że nie korzystałem ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję opiekunowi pracy, Panu Doktorowi Markowi Długoszowi, a także opiekunowi pracowni inżynierskiej, Panu Doktorowi Januszowi Tenecie, za niesioną pomoc i zawsze dobrą radę.

Spis treści

1. Wstęp

Laboratorium pojazdów autonomicznych AGH–Delphi ma za zadanie zbudowanie pojazdu autonomicznego na bazie udostępnionego pojazdu elektrycznego (zob. zdjęcie 1.1). Podczas prac konieczne jest zapewnienie bezpieczeństwa podczas testowania algorytmów sterowania. Algorytmy takie mogą nie działać poprawnie, w związku z czym w trakcie testów może zajść potrzeba awaryjnego przerwania jazdy samochodu.

Fig. 1.1. Samochód elektryczny udostępniony przez firmę Delphi (zdjęcie: P. Banaszkiewicz).

Awaryjne przerwanie jazdy samochodu powinno odbyć się poprzez wyłączenie zasilania silnika samochodu. O ile jest to możliwe, należy uniknąć odłączenia zasilania reszty samochodu (np. układu kierowniczego lub komputera pokładowego).

1.1. Cele projektu

System ESAV (ang. *Emergency Stop Autonomous Vehicle* – zdalny wyłącznik bezpieczeństwa dla samochodu autonomicznego) powinien charakteryzować się następującymi właściwościami:

– Możliwość natychmiastowego zatrzymania pojazdu na polecenie operatora.

– Natychmiastowe zatrzymanie pojazdu w przypadku utraty zasilania systemu ESAV.

– Natychmiastowe zatrzymanie pojazdu w przypadku przerwania transmisji pomiędzy modułami systemu ESAV.

– Natychmiastowe zatrzymanie pojazdu w przypadku zakłóceń pracy systemu ESAV (na przykład w wyniku błędów transmisji).

– Możliwość poruszania się pojazdu tylko jeśli system ESAV jest zasilany.

– Możliwość poruszania się pojazdu tylko w przypadku rozpoczęcia prawidłowej transmisji danych

– pomiędzy modułami systemu ESAV.

– Możliwość wyświetlenia aktualnego stanu systemu ESAV.

W niniejszym opisie projektu używane będą następujące pojęcia skrótowe:

– **moduł A**: bazowy moduł systemu ESAV; to z niego korzysta operator aby wyłączyć zasilanie silnika w samochodzie,

– **moduł B**: moduł montowany w samochodzie, który odpowiedzialny jest za wyłączenie zasilania silnika,

– **tryb awaryjny**: tryb modułu B w którym następuje odłączenie zasilania silnika samochodu.

Zasilanie obydwu modułów powinno być całkowicie niezależne od zasilania pojazdu, a wymianę akumulatorów łatwa. Ponadto obydwa moduły muszą monitorować i pokazywać poziom naładowania swoich akumulatorów.

Po wyłączeniu pojazdu przez system ESAV układ musi zostać przywrócony do pracy poprzez zresetowanie.

System ESAV powinien być systemem bezobsługowym: raz skonfigurowany ma poprawnie pracować bez ingerencji użytkownika. Budowa systemu powinna umożliwiać innym osobom kontynuowanie prac nad nim. W tym celu jego oprogramowanie powinno być otwarto-źródłowe, a sprzęt wymienialny.

1.2. Wymagania funkcjonalne

Na podstawie opisu z rozdziału 1.1 wyznaczono poniższe cele funkcjonalne systemu ESAV:

1.2.1. Moduł B przy braku zasilania ma odcinać zasilanie silnika samochodu.

1.2.2. Po włączeniu moduł B ma oczekiwać na dane aktywujące jego pracę, które ma otrzymać od modułu A. Stan modułu B sygnalizowany jest migającą czerwoną diodą.

1.2.3. Dopóki moduł B nie odbierze danych aktywujących, pojazd nie może się poruszać.

1.2.4. Po odebraniu danych aktywujących, moduł B powinien umożliwić jazdę samochodu. Stan modułu B sygnalizowany jest świecącą zieloną diodą.

1.2.5. Jeżeli podczas działania moduł B przestanie odbierać transmisję, musi on przejść w tryb awaryjny. Stan modułu sygnalizowany jest ciągle świecącą się czerwoną diodą.

1.2.6. Po naciśnięciu przycisku wyłączenia awaryjnego w module A przez operatora, moduł B musi przejść w tryb awaryjny. Stan modułu sygnalizowany jest ciągle świecącą się czerwoną diodą.

1.2.7. Po zresetowaniu modułu B poprzez naciśnięcie przycisku „Reset”, moduł B powraca z trybu awaryjnego do stanu oczekiwania na dane aktywujące. Stan modułu sygnalizowany jest migającą czerwoną diodą.

1.2.8. Po wyłączeniu zasilania modułu B musi nastąpić odłączenie zasilania silnika samochodu.

1.2.9. Bateria (akumulator) modułu B może być ładowana w trakcie ładowania akumulatorów pojazdu.

1.3. Wymagania użytkowe

Na podstawie opisu z rozdziału 1.1 wyznaczono poniższe cele zasięgu, czasu oraz szybkości działania systemu ESAV:

1.3.1. System ESAV musi pracować bez konieczności wymiany bądź ładowania baterii (akumulatorów) przez minimum 5 godzin.

1.3.2. System ESAV musi być w stanie co 1 sekundę odbierać i dekodować dane, a następnie je wysyłać.

1.3.3. System ESAV musi pracować poprawnie na dystansie 100 metrów w otwartej przestrzeni.

1.3.4. System ESAV musi pracować poprawnie w pomieszczeniach.

1.4. Wymagania współdziałania

Na podstawie opisu z rozdziału 1.1 wyznaczono poniższe cele integracji systemu ESAV z innymi urządzeniami:

1.4.1. Moduł B musi poprawnie wyłączać silnik samochodu.

1.4.2. Moduł B powinien mieć układ ładujący podłączony do systemu elektrycznego pojazdu.

1.5. Wymagania bezpieczeństwa

Na podstawie opisu z rozdziału 1.1 wyznaczono poniższe wymagania bezpieczeństwa przy korzystaniu z systemu ESAV:

1.5.1. Moduł B systemu ESAV nie może doprowadzić do porażenia użytkownika prądem.

1.5.2. Komunikacja pomiędzy modułami powinna zabezpieczać przed atakami powtórzenia poleceń operatora.

1.5.3. Moduł B systemu ESAV nie może nagrzewać się powyżej temperatury $50\,^{\circ}\mathrm{C}$.

1.6. Wymagania pochodne

Wymagania, które nie zostały jawnie wspomniane w rozdziałach 1.2—1.5, a wynikają albo z opisu projektu (rozdział 1.1), albo pojawiły się w momencie implementowania projektu, zostały opisane w tym rozdziale.

Moduł B musi:

1.6.1. monitorować poziom naładowania (napięcie) na baterii zasilającej,

1.6.2. sygnalizować włączenie trybu awaryjnego świeceniem czerwonej diody,

1.6.3. sygnalizować oczekiwanie na transmisję miganiem czerwonej diody,

1.6.4. sygnalizować poprawne działanie świeceniem zielonej diody.

Moduł A musi:

1.6.5. monitorować poziom naładowania (napięcie) na baterii zasilającej,

1.6.6. sygnalizować wysyłanie informacji miganiem zielonej diody,

1.6.7. mieć małe wymiary (aby możliwe było trzymania go w dłoniach).

Komunikacja między modułami powinna umożliwiać wymianę takich informacji:

1.6.8. PING: wymiana danych co około sekundę (do weryfikacji istnienia połączenia),

1.6.9. STOP: do wysłania danych w celu natychmiastowego przejścia modułu B w tryb awaryjny,

1.6.10. DATA: do wymiany danych diagnostycznych (np. napięcia baterii).

1.7. Podsumowanie

Podejście do projektu oparte o listę wymagań (ang. *requirements-driven design* [1] albo *feature-driven development* [2]) umożliwia sprecyzowanie niewielkich wymagań a priori, które pozwalają wykonawcy implementować funkcjonalności pojedyncze i skierowane na konkretne wymagania, a także śledzić spełnienie tych wymagań w formie prostej tabeli (zob. rozdział 2.3).

W tym rozdziale zostały wyszczególnione wymagania według sugerowanego schematu z [1], jednakże w przypadku projektów stricte programistycznych schemat ten może wyglądać inaczej.

Skupienie się na wyszczególnieniu wymagań przed rozpoczęciem prac nad projektem znacząco ułatwiło jego tworzenie, m.in. poprzez eliminację luk w opisie funkcjonalnym systemu ESAV.

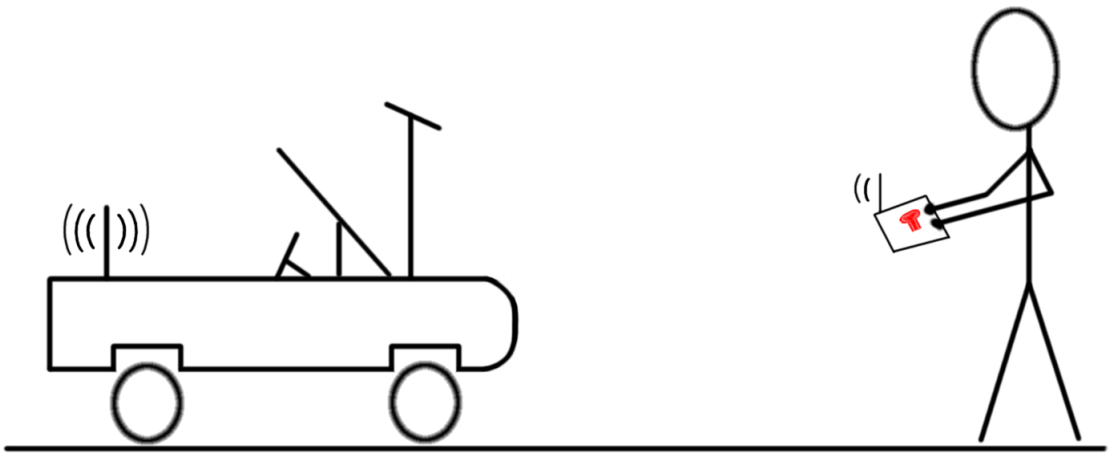

Fig. 1.2. Rysunek poglądowy systemu w trakcie działania (rysenek: P. Banaszkiewicz).

2. Wykonanie projektu

W tym rozdziale opisane zostało wykonanie projektu zgodnie z wymaganiami postawionymi w rozdziale 1.

2.1. Opis części sprzętowej

W poniższym rozdziale została opisana platforma sprzętowa modułów A i B oraz ich współdziałanie z innymi komponentami, m.in. z zasilaniem akumulatorowym czy z przekaźnikiem w samochodzie elektrycznym.

2.1.1. Moduł A

Jest to moduł bazowy, tj. trzymany przez operatora w dłoni. Ze względu na użycie Arduino Pro 328 5V 16MHz [3] (zob. zdjęcie 2.1) jako platformy sprzętowej, ma on niewielkie wymiary.

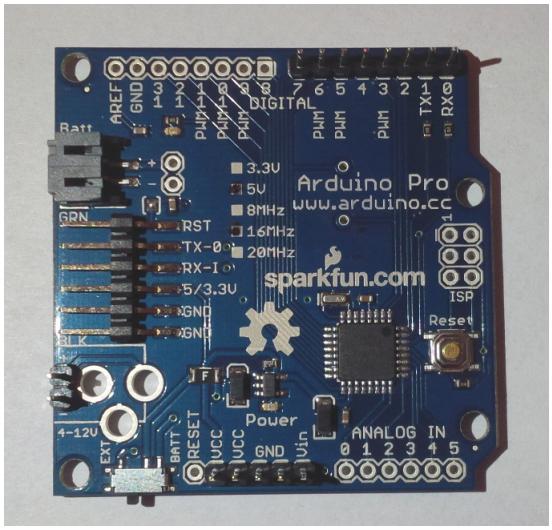

Fig. 2.1. Platforma sprzętowa Arduino Pro 328 5V 16MHz (zdjęcie: Piotr Banaszkiewicz).

2.1.2. Moduł B

Jest to moduł, który został zamontowany w samochodzie. Jako platformę użyte zostało ponownie Arduino Pro 328 5V 16MHz [3] (zob. zdjęcie 2.1). Podstawowa funkcja, tj. odłączanie i załączanie silnika elektrycznego, odbywa się za pomocą przekaźnika ASR-90DD-H firmy Anly Electronics (zob. zdjęcie 2.2 i rozdział 2.1.7).

Fig. 2.2. Przekaźnik półprzewodnikowy ASR-90DD-H (zdjęcie: Piotr Banaszkiewicz).

2.1.3. Zasilanie

Obydwa moduły sprzętowe są zasilane z powerbanków Solo5 10000 mAh firmy Romoss. Posiadają one wbudowane różne zabezpieczenia, m.in.: natężeniowe (dla zachowania żywotności), temperaturowe czy zwarciowe (ang. *short-circuit*). Mają również układ, który wyłącza powerbank w sytuacji, gdy prąd ładowania jest bardzo niewielki [4].

Moduły A i B pobierają bardzo niewiele prądu (zmierzone: między $20\,\text{mA}$ a $50\,\text{mA}$), więc aby powstrzymać powerbanki przed rozłączaniem zwarto masę i źródło energii dwunastoma (w przypadku modułu A) lub trzynastoma (w przypadku modułu B) rezystorami $910\,\Omega$ wpiętymi równolegle.

Połączenie równoległe takiej ilości rezystorów daje dość niską rezystancję zastępczą (teoretycznie: $75{,}83\,\Omega$ dla modułu A, $70\,\Omega$ dla modułu B; praktycznie: występują różnice w faktycznych wartościach rezystancji rezystorów, zwłaszcza gdy te się nagrzeją) i pozwala na obniżenie wydzielanego ciepła na rezystorach, co skutkuje:

– większym poborem prądu, dzięki czemu powerbanki się nie wyłączają,

– nagrzewaniem rezystorów (z prawa Ohma: $P = \frac{U^2}{R} = \frac{(5.1)^2}{900} = 0.0289[W]$, gdzie $5{,}1\,\text{V}$ to zmierzone napięcie na rezystorze, a $900\,\Omega$ to przybliżona rezystancja nagrzanego rezystora), a więc w

przypadku 12 rezystorów położonych bardzo blisko siebie mamy około $0{,}34$ W wydzielanej mocy w tym miejscu.

Ponadto banki energii umożliwiają jednoczesne ładowanie jak i rozładowywanie, tzn. jest możliwe wpięcie źródła zasilania na wejście powerbanku, jak i obciążenia na jego wyjście. Dzięki temu powerbank zasilający moduł B można ładować z portu USB znajdującego się w samochodzie.

Sama pojemność $10\,000$ mA h teoretycznie[1] pozwala na zasilanie układu pobierającego 100 mA przez 100 godzin lub układu pobierającego 200 mA przez 50 godzin. Każdy z modułów A i B po obciążeniu przez rezystory pobierał między 100 a 200 mA. Poziom naładowania powerbanków można obserwować na ich diodowych wskaźnikach.

2.1.4. Komunikacja

Do komunikacji wykorzystano parę modułów radiowych HC-12 433 MHz [5], które łączą się z platformami sprzętowymi za pomocą łącza szeregowego. Prędkość komunikacji takiego modułu w powietrzu wynosi 5000bps, czyli typowe pakiety przesyłane przez system (zob. rozdział 2.2.2), mające rozmiar $4B = 32b$ są przesyłane w ułamku sekundy.

Zasięg komunikacji modułów został przetestowany i wynosi 25 m w pomieszczeniu (przy braku ścian pomiędzy modułami) oraz 100 m na otwartej przestrzeni. Gdyby odległości te były niewystarczające, HC-12 umożliwia podpięcie silniejszej zewnętrznej anteny [5] na złączu U.FL.

2.1.5. Obudowy

Dla każdego modułu systemu ESAV przygotowano sztywną nakładkę w formie płytki prototypowej (zdjęcia 2.3 i 2.4) z przylutowanymi komponentami wykorzystywanymi przez dany moduł. Zakładane są one na górne i dolne piny modułów A i B (zob. zdjęcie 2.1).

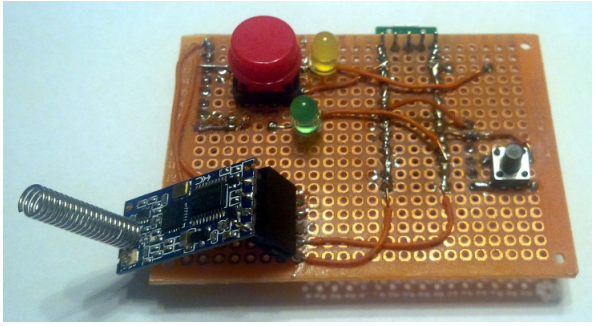

Fig. 2.3. Nakładka na moduł A (zdjęcie: Piotr Banaszkiewicz).

Ponadto na każdą platformę została zaprojektowana obudowa. Na zdjęciu 2.1 widać cztery otwory w platformie Arduino (dwa przy lewej krawędzi płytki PCB oraz dwa przy prawej krawędzi), które zostały użyte do przykręcenia każdej z platform do obudowy.

[1] W praktyce pojemność $10\,000$ mA h jest nieosiągalna, gdyż powerbanki posiadają mechanizmy wydłużające ich żywotność; polega to na ładowaniu akumulatorów powerbanku tylko do pewnego stopnia.

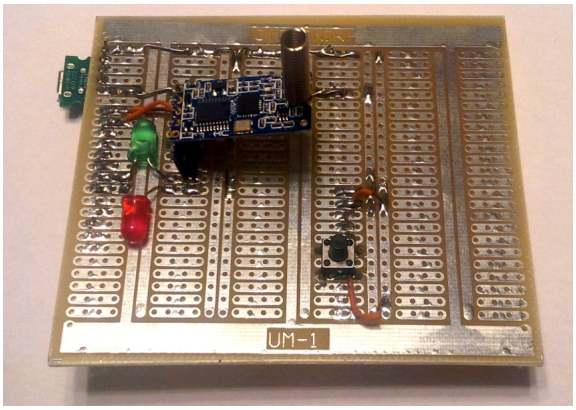

Fig. 2.4. Nakładka na moduł B (zdjęcie: Piotr Banaszkiewicz).

Obudowy bazują na istniejącym projekcie kontenerów dla Arduino Leonardo oraz Uno wykonanym przez Zygmunta Wojcika [6]. Ich licencja zezwala na modyfikacje (więcej szczegółów dostępnych jest pod adresem [7]).

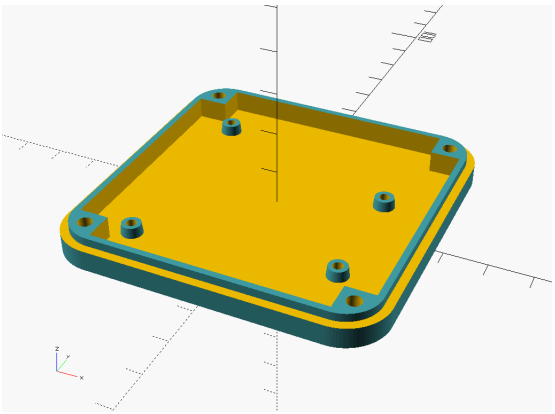

Fig. 2.5. Obudowa modułu A (oryginalny autor: Zygmunt Wojcik, modyfikacje: Piotr Banaszkiewicz [7]).

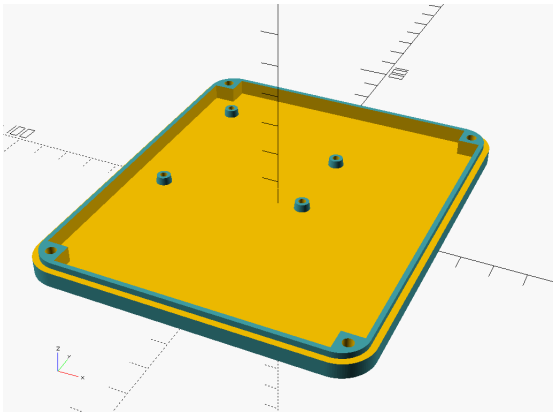

Fig. 2.6. Obudowa modułu B (oryginalny autor: Zygmunt Wojcik, modyfikacje: Piotr Banaszkiewicz [7]).

2.1.6. Schematy połączeń

Symboliczny sposób połączenia modułu A, wejścia zasilającego USB oraz transceivera radiowego został przedstawiony na schemacie 2.7.

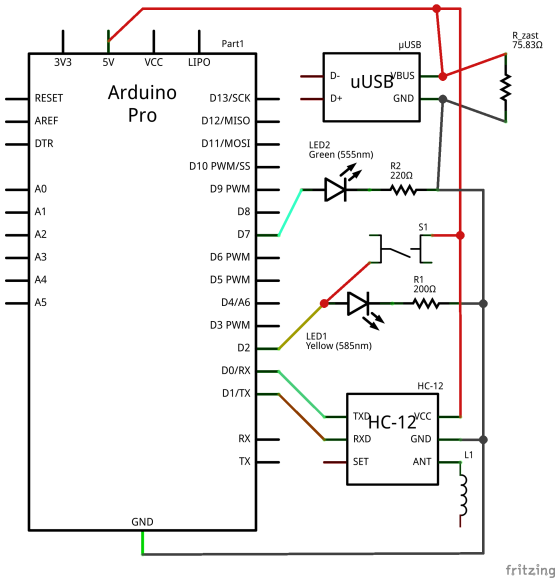

Fig. 2.7. Schemat połączeń modułu A wygenerowany w programie Fritzing [8] (opracowanie własne).

Bardzo podobnie wygląda symboliczny schemat połączeń modułu B, z tym że on dodatkowo steruje przekaźnikiem (schemat 2.8).

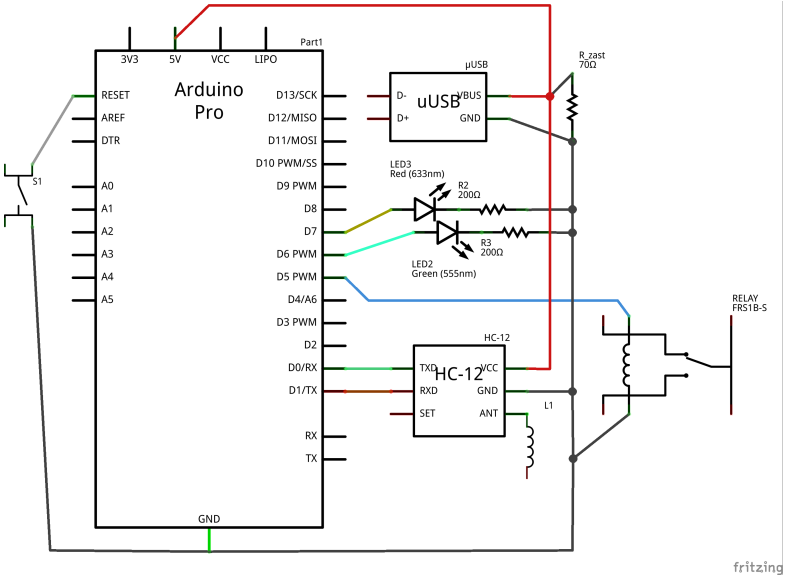

Fig. 2.8. Schemat połączeń modułu B wygenerowany w programie Fritzing [8] (opracowanie własne).

2.1.7. Odłączanie i załączanie silnika samochodu

Pomiędzy kabel zasilający a moduł sterujący pracą silnika samochodu został wpięty przekaźnik ASR-90DD-H (zob. zdjęcie 2.2). Moduł sterujący odpowiedzialny jest za przetworzenie napięcia stałego na prąd zmienny poprowadzony bezpośrednio na uzwojenia silnika.

Przekaźnik ASR-90DD-H jest to przekaźnik półprzewodnikowy, a zatem typu *Normally Open*, sterowany napięciem stałym 3–32 V. Przekaźnik jest sterowany wyjściem z modułu B (zob. element „RELAY" na schemacie 2.8).

Według danych producenta, przekaźnik jest w stanie obsłużyć na wyjściu mocy do 90 A prądu stałego pod napięciem do 240 V. Wybrano przekaźnik mocno przekraczający spodziewane natężenie prądu, gdyż zapewnia to mniejsze wydzielanie ciepła (zob. rozdział 2.1.8).

Niestety producent nie udostępnia schematu elektrycznego przekaźnika w dokumentacji [9], więc podczas jego montażu zadbano o kilka elementów bezpieczeństwa:

– zastosowano podkładkę silikonową jako izolator elektryczny,

– zastosowano radiator chłodzący (widoczny na zdjęciu 2.9).

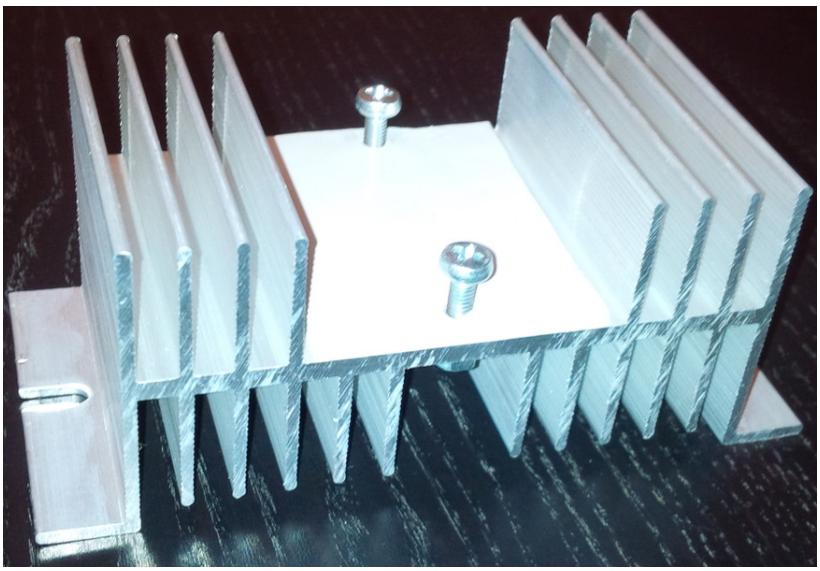

Fig. 2.9. Radiator chłodzący przekaźnik SSR (zdjęcie: Piotr Banaszkiewicz).

Elementy te zgodne są z zaleceniami polskiego producenta przekaźników, firmy Relpol [10].

2.1.8. Chłodzenie przekaźnika półprzewodnikowego

Przekaźniki półprzewodnikowe z powodu pewnej niskiej oporności własnej w momencie podłączonego wyjścia mocy wydzielają dużą ilość energii w postaci ciepła.

W ramach pracy inżynierskiej przeprowadzono eksperyment polegający na zmierzeniu temperatury obudowy działającego przekaźnika podczas włączonego obciążenia.

Przekaźnik został umieszczony na radiatorze i pracował przez kilka minut (do momentu ustalenia się temperatury) pod stałym obciążeniem. Kamera termowizyjna ostawiona była tak, by mierzyć temperaturę radiatora oraz obudowy przekaźnika. Tabela 2.1 zawiera zestawienie długości czasu działania przekaźnika pod obciążeniem, a także uzyskanej na końcu eksperymentu temperatury, natomiast zdjęcia 2.10 i 2.11 ukazują obrazy z kamery termowizyjnej.

Tabela 2.1. Parametry eksperymentu obciążenia przekaźnika półprzewodnikowego

Czas obciążenia	Napięcie	Natężenie	Temperatura radiatora[a]	Temperatura obudowy przekaźnika[a]
8 min	45 V	36 A	25 °C	ok. 60 °C
8 min	42,5 V	43 A	25–35 °C	ok. 90 °C

[a] pod koniec pomiaru

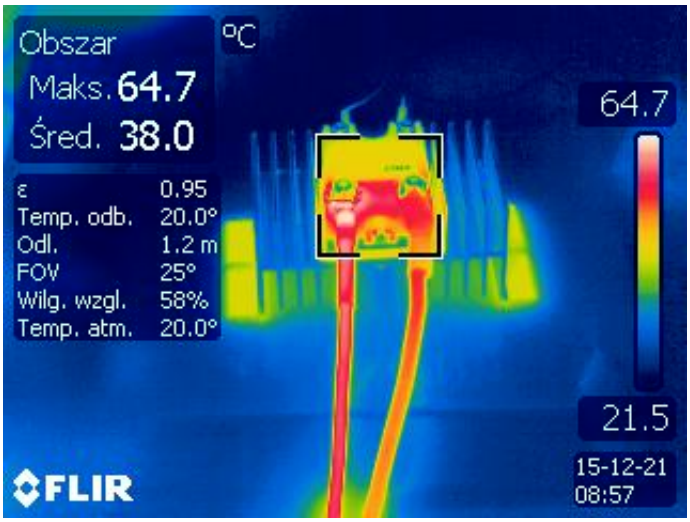

Fig. 2.10. Obraz uzyskany za pomocą kamery termowizyjnej na koniec trwania eksperymentu z mniejszym obciążeniem (zdjęcie: Marek Długosz).

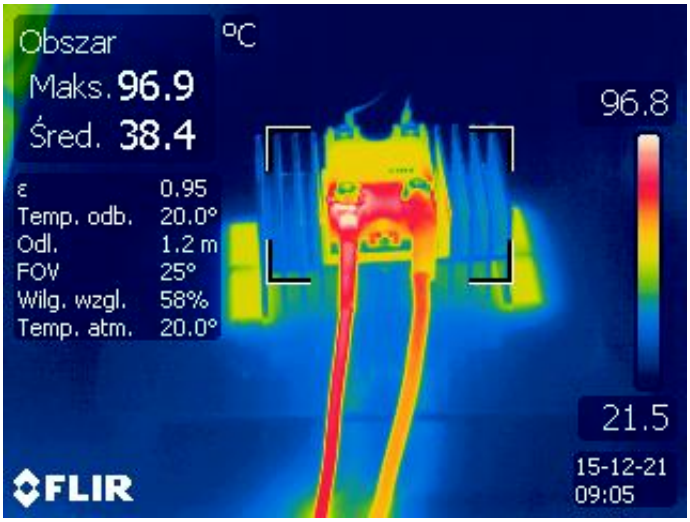

Fig. 2.11. Obraz uzyskany za pomocą kamery termowizyjnej na koniec trwania eksperymentu z większym obciążeniem (zdjęcie: Marek Długosz).

Na powyższych zdjęciach najcieplejszym elementem była śruba służąca do przykręcenia kabla zasilającego. Temperatura ciepłego miejsca na obudowie przekaźnika była o około 15 °C niższa; uwzględnione to zostało w tabeli 2.1 poprzez odjęcie ok. 10 °C od maksymalnej mierzonej temperatury.

Ponadto należy zauważyć jak niewiele nagrzewał się radiator, do którego przymocowany został przekaźnik. To udowadnia, że jest on wystarczający dla zastosowania w samochodzie.

Na samym końcu należy zauważyć, że przekaźnik nie powinien być w realnych zastosowaniach narażony na tak duże obciążenia przez tak długi czas; według obserwacji, maksymalny prąd chwilowy w trakcie przyspieszania pojazdu i skręcania jednocześnie wynosił około 50 A.

2.2. Opis części programowej

W tym rozdziale opisano m.in. logikę, algorytmy i struktury danych wykorzystywane przez programy modułów A i B.

2.2.1. Schematy cykli-działań

Poniższe schematy reprezentują logikę, zdarzenia oraz czynności, jakimi posługują się moduły A i B.

Dokładny opis pakietów danych oraz momenty, w których są przesyłane, znajduje się w rozdziale 2.2.2.

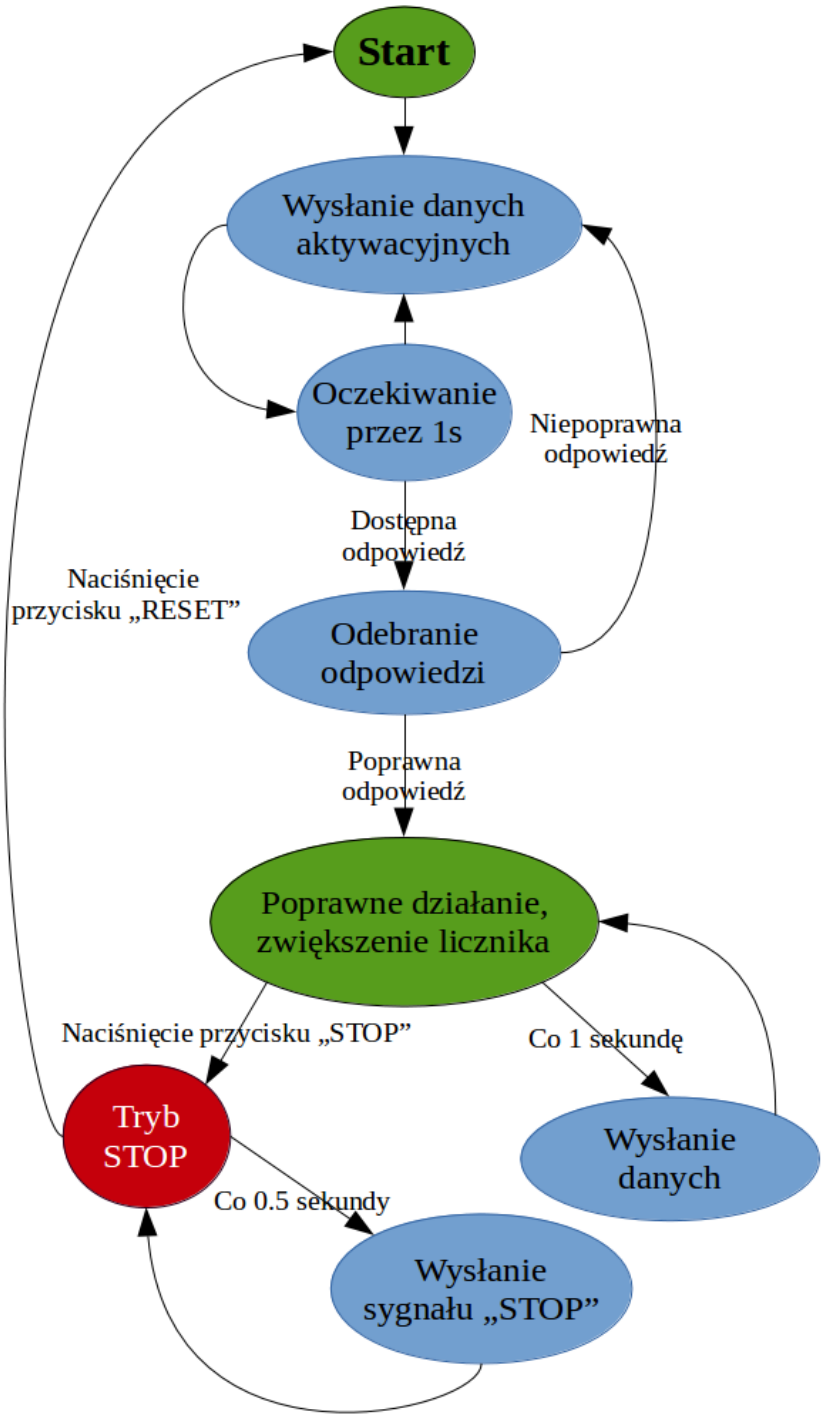

Fig. 2.12. Graf cykli-działań dla modułu A (opracowanie własne).

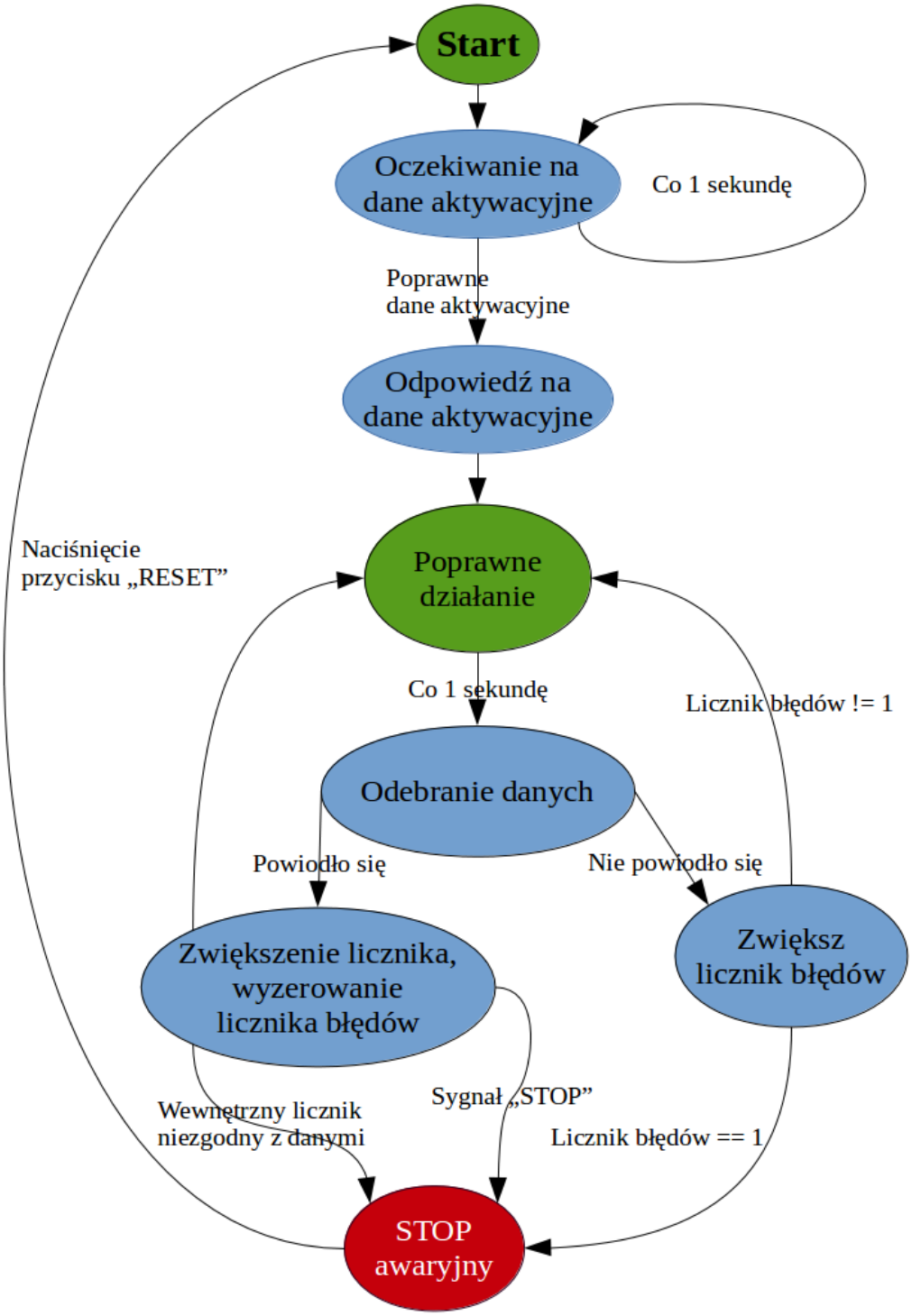

Fig. 2.13. Graf cykli-działań dla modułu B (opracowanie własne).

2.2.2. Format pakietów

Każdy pakiet ma długość 4 bajtów:

$$0\text{x}\underbrace{\text{FC}}_{\text{e}}\underbrace{\text{ABCD}}_{\text{c}}\underbrace{00}_{\text{s}} \tag{2.1}$$

gdzie

- e – bajt *emergency*. Jeżeli jego wartość to $0xFF$ to moduł B ma natychmiast przejść do trybu awaryjnego. W praktyce oznacza to, że operator wcisnął przycisk „STOP" modułu A.
- c – 2 bajty zapisujące wartość liczbową licznika modułu A (typ bez znaku, 16 bitów),
- s – bajt stopu, jego wartość zawsze ma być równa $0x00$.

Istnieją 2 pakiety specjalne przesyłane między modułami A i B w celu synchronizacji i rozpoczęcia działania:

- pakiet danych aktywacyjnych wysyłany przez moduł A: $0\text{x}\text{FFFFFF}00$,
- pakiet odpowiedzi na dane aktywacyjne wysyłany przez moduł B: $0\text{x}11\text{FFFF}00$,
- ponadto każdy pakiet danych wysyłany przez moduł A już po odebraniu odpowiedzi na dane aktywacyjne zawiera bajt *emergency* ustawiony na wartość $0\text{x}22$ – za wyjątkiem sytuacji, w której moduł A żąda od modułu B przejścia w tryb awaryjny; wtedy ten bajt ustawiony jest na wartość $0\text{x}\text{FF}$.

Przykładowe pakiety:

- $0\text{x}223\text{F}0\text{E}00$: moduł A wysyła informację o braku konieczności przejścia w tryb awaryjny oraz o wartości licznika (16142),
- $0\text{x}\text{FFA}37800$: moduł A wysyła informację o przejściu w tryb awaryjny oraz o ostatniej wartości licznika (41848).

2.2.3. Maszyna stanów

Oprogramowanie modułów A i B wykorzystuje proste maszyny stanów [11].
Stany modułu A:

1. CONNECTING: wysyłanie pakietu danych aktywacyjnych co 1 sekundę, oczekiwanie na odpowiedź,
2. PING: wysyłanie obecnej wartości licznika co 1 sekundę, zwiększanie wartości licznika,
3. STOP: wysyłanie ostatniej wartości licznika oraz ustawionego bajtu *emergency* co $0{,}5$ sekundy.

Stany modułu B:

1. WAIT: oczekiwanie na przyjście pakietu danych aktywacyjnych; odesłanie na niego odpowiedzi,

2. RUN: zwiększanie wewnętrznego licznika, odbieranie wartości licznika modułu A, porównywanie obu liczników, sprawdzanie bajtu *emergency*,
3. STOP: podtrzymywanie wyłączenia zasilania.

Warunki przejścia między stanami modułu A:

– CONNECTING → PING: gdy pakiet odpowiedzi na dane aktywacyjne wysłany przez moduł B to $0\text{x}11\text{FFFF}00$,

– PING → STOP: gdy operator wciśnie przycisk „RESET”.

Warunki przejścia między stanami modułu B:

– WAIT → RUN: gdy pakiet danych aktywujących wysłany przez moduł A to $0\text{xFFFFFF}00$,

– RUN → STOP: gdy jeden z warunków jest spełniony:

  – bajt *emergency* w przychodzącym pakiecie ma wartość 0xFF,
  – licznik wewnętrzny ma wartość różną od wartości licznika wysłanej przez moduł A w przychodzącym pakiecie,
  – przez 3 sekundy nie nadeszły żadne dane od modułu A.

2.2.4. Panel operatorski

Obydwa moduły systemu ESAV zawierają minimalny interfejs dla użytkownika. Składa się on odpowiednio z:

– dla modułu A:

  – przycisk „STOP” wywołujący przerwanie wewnętrzne platformy Arduino i w konsekwencji wysłanie polecenia przejścia do trybu awaryjnego,
  – przycisk „RESET” do łatwiejszego restartu urządzenia,
  – zieloną diodę informującą o wysyłaniu danych przez port szeregowy,
  – żółtą diodę włączającą się w momencie naciśnięcia przycisku „STOP” (spełnia ona funkcję wizualnego sprzężenia zwrotnego dla użytkownika),

– dla modułu B:

  – przycisk „RESET” do łatwiejszego restartu urządzenia,
  – zieloną diodę informującą o poprawnym trybie pracy (stan „RUN”),
  – czerwoną diodę informującą o oczekiwaniu na dane wejściowe (szybka pulsacja) lub o przejściu w tryb awaryjny (ciągłe świecenie).

2.2.5. Szyfrowanie komunikacji

Komunikacja pomiędzy modułami (za wyjątkiem przekazywanych danych aktywacyjnych i odpowiedzi na nie) jest w prosty sposób szyfrowana.

Do szyfrowania wykorzystywane są dwie tablice, każda o długości 256, co odpowiada ilości wartości pojedynczego bajtu. Szyfrowanie pojedynczego bajtu polega na odszukaniu w tablicy szyfrującej odpowiadającej mu wartości zaszyfrowanej, a odszyfrowanie polega na odszukaniu w tablicy deszyfrującej wartości oryginalnej odpowiadającej wartości zaszyfrowanej.

Poniższy przykład ilustruje szyfrowanie i deszyfrowanie na przykładzie czterech wartości:

Listing 2.1. Pseudokod zbliżony do C pokazujący zasadę szyfrowania i deszyfrowania komunikacji systemu ESAV.

```
byte cipher_array[] = {3, 1, 0, 2};
byte decipher_array[] = {2, 1, 3, 0};
byte original = 2;
byte encrypted = cipher_array[original];
assert(original == decipher_array[encrypted]);
```

Dzięki wykorzystaniu tej prostej metody nie nastąpił żaden narzut w wielkości przesyłanych pakietów, niestety skutkiem słabszego bezpieczeństwa niż w przypadku stosowania np. algorytmu AES [12].

2.2.6. Programowanie Arduino

Platforma Arduino, na której oparty jest system ESAV, programowana jest w środowisku Arduino IDE (zob. 2.14) za pomocą języka bardzo zbliżonego do C. Opis jego składni, udostępnionych funkcji i klas dostępny jest w [13].

Podczas pracy nad projektem stworzono dwa pliki źródłowe, odpowiednio dla modułu A oraz B, a także jedną bibliotekę w C++ współdzieloną przez oba projekty, której zadaniem jest obsługiwanie szyfrowania i deszyfrowania komunikacji pomiędzy modułami.

Pliki źródłowe dostępne są w serwisie GitHub [14].

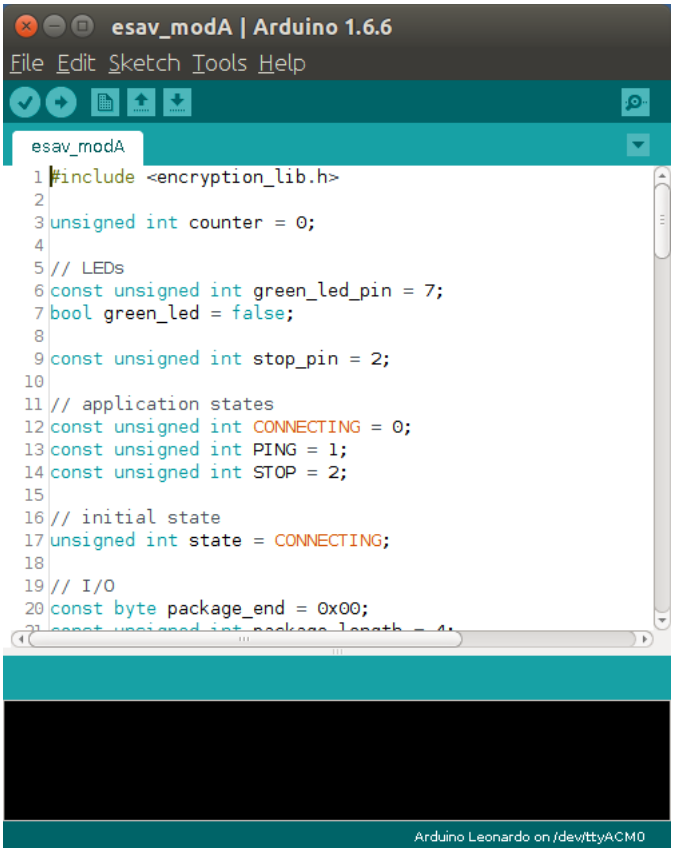

Fig. 2.14. Interfejs programu Arduino IDE używany do programowania na platformę Arduino (zdjęcie: Piotr Banaszkiewicz).

2.2.7. Kabel do programowania Arduino

Platforma Arduino Pro, w przeciwieństwie do innych większych mikrokontrolerów Arduino, nie zawiera programatora na USB. Programowanie Arduino Pro odbywa się poprzez port szeregowy (6 pinów usytuowanych równolegle do płytki PCB, widocznych po lewej stronie na zdjęciu 2.1).

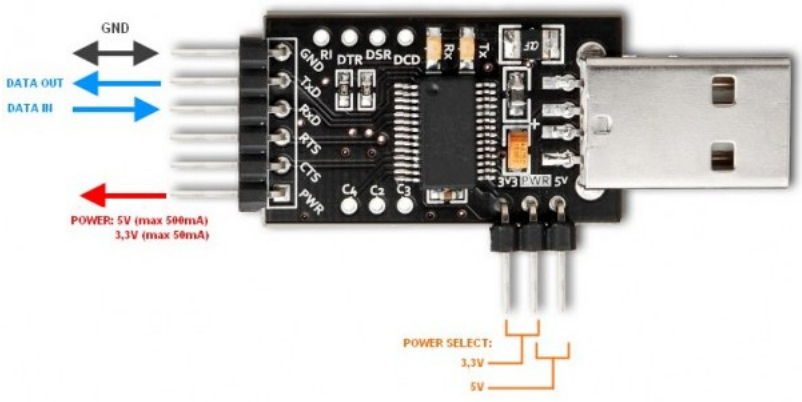

Fig. 2.15. Opis wyjść używanego konwertera USB-UART (zdjęcie: Botland [15]).

Niestety używany konwerter USB-UART posiada zgodne wyprowadzenia (zob. 2.15), ale w nieodpowiedniej kolejności, co zostało przedstawione w tabeli 2.2.

Tabela 2.2. Złącza komunikacji szeregowej i zasilania platformy Arduino Pro oraz używanego konwertera USB-UART.

Wyprowadzenie Arduino Pro	Wyprowadzenie konwertera USB-UART
RST	GND
TX	TXD
RX	RXD
5V	RTS
GND	CTS
GND	PWR

Podłączenia wymagane do prawidłowego zaprogramowania Arduino Pro:

– RST → RTS

– TX → RXD

– RX → TXD

– 5V → PWR

– GND → GND

W związku z tym konieczne stało się stworzenie kabla pozwalającego łatwo programować moduły systemu ESAV. Został on zbudowany na podstawie tzw. skrętki ethernetowej (zdjęcie 2.16).

Fig. 2.16. Zdjęcie kabla wykorzystywanego do programowania Arduino (zdjęcie: Piotr Banaszkiewicz).

2.3. Zakres zrealizowanych wymagań

W poniższej tabeli zestawiono założenia projektu wyszczególnione w rozdziale wraz z informacją o ich spełnieniu (bądź wytłumaczeniem dlaczego nie zostały spełnione) oraz odnośnikiem do rozdziału zawierającego bardziej szczegółowe informacje na temat danej funkcjonalności.

Tabela 2.3. Macierz śledzenia wymagań (ang. *requirements traceability matrix* [1]).

Numer wymagania	Wymagane?	Spełnione?	Jak?	Roz.
1.2.1.	Tak	Tak	Przekaźnik typu *Normally Open*.	2.1.7
1.2.2.	Tak	Tak	Oprogramowanie modułów A i B.	2.2.1
1.2.3.	Tak	Tak	Przekaźnik typu *Normally Open* oraz oprogramowanie modułu B.	2.2.3
1.2.4.	Tak	Tak	Oprogramowanie modułu B.	2.2.3
1.2.5.	Tak	Tak	Oprogramowanie modułu B.	2.2.3
1.2.6.	Tak	Tak	Oprogramowanie modułów A i B.	2.2.3
1.2.7.	Tak	Tak	Podpięcie przycisku „RESET” do pinu RESET modułu B.	2.2.4
1.2.8.	Tak	Tak	Przekaźnik typu *Normally Open*.	2.1.7
1.2.9.	Nie	Nie	Zakupione powerbanki umożliwiają ładowanie podczas używania ich, jednakże to w gestii użytkownika jest korzystanie z tej możliwości.	2.1.3
1.3.1.	Tak	Tak	Powerbanki powinny teoretycznie zapewnić zasilanie przez około 50–100 godzin.	2.1.3
1.3.2.	Tak	Tak	Prędkość przesyłania danych między modułami HC-12 wynosi 5000bps.	2.1.4
1.3.3.	Tak	Tak	Zasięg 100 m na otwartej przestrzeni.	2.1.4
1.3.4.	Tak	Tak	Zasięg 25 m w pomieszczeniach (przy braku ścian pomiędzy modułami).	2.1.4
1.4.1.	Tak	Tak	Moduł B odłącza zasilanie modułu, który bezpośrednio zasila silnik samochodu.	2.1.7

1.4.2.	Nie	Nie	Zakupione powerbanki umożliwiają ładowanie podczas używania ich, jednakże to w gestii użytkownika jest korzystanie z tej możliwości.	2.1.3
1.5.1.	Tak	Tak	Wejście sterowania przekaźnika jest optoizolowane.	2.1.7
1.5.2.	Tak	Tak	Komunikacja pomiędzy modułami polega na wymianie szyfrowanego licznika.	2.2.5
1.5.3.	Tak	Tak	Przekaźnik dopiero wystawiony na bardzo długie działanie pod wysokim obciążeniem może osiągnąć temperaturę 100 °C.	2.1.8
1.6.1.	Nie	Nie	Poziom naładowania jest wyświetlany na obudowie powerbanku, więc nie ma konieczności dodatkowego monitorowania przez moduł B.	2.1.3
1.6.2.	Tak	Tak	Spełnione dosłownie.	2.2.4
1.6.3.	Tak	Tak	Spełnione dosłownie.	2.2.4
1.6.4.	Tak	Tak	Spełnione dosłownie.	2.2.4
1.6.5.	Nie	Nie	Poziom naładowania jest wyświetlany na obudowie powerbanku, więc nie ma konieczności dodatkowego monitorowania przez moduł B.	2.1.3
1.6.6.	Tak	Tak	Spełnione dosłownie.	2.2.4
1.6.7.	Tak	Tak	Spełnione dosłownie.	2.1.5
1.6.8.	Tak	Tak	Wysłanie pakietu danych jest rozumiane jako „PING”.	2.2.2
1.6.9.	Tak	Tak	Opisane w formacie pakietów.	2.2.2
1.6.10.	Nie	Nie	Niewymagane.	

2.4. Podsumowanie

Wykonanie projektu zostało podzielone na dwie części: część związaną stricte ze sprzętem oraz część związaną z programowaniem. Umożliwiło to zrównoleglenie pracy: w momencie oczekiwania na sprzęt rozwijana było oprogramowanie systemu ESAV.

Dzięki opisaniu wszystkich wymagań w rozdziale 1, możliwe zostało ułożenie tabeli 2.3, która od razu pokazuje w jakim stanie ukończenia jest projekt.

3. Podsumowanie projektu

Celem niniejszej pracy inżynierskiej było stworzenie narzędzia ułatwiającego bezpieczne testowanie elektrycznego samochodu autonomicznego w laboratorium AGH–Delphi.

Za wkład własny w pracę uważam zrealizowanie następujących zagadnień:

- zebranie wymagań (któremu poświęcono prawie cały rozdział 1),
- stworzenie pierwszego prototypu systemu (nieopisanego w tej pracy, bazującego na architekturze Raspberry Pi i sieci WiFi),
- opracowanie schematów logicznych oraz formatu komunikacji między modułami (2.2.1, 2.2.2),
- zbudowanie drugiego prototypu systemu, już przy wykorzystaniu platformy docelowej Arduino i płytek wielostykowych do podłączenia elementów panelu operatora oraz modułów komunikacyjnych,
- wykonanie kabla pozwalającego programować Arduino (2.2.7),
- zaimplementowanie algorytmów na podstawie opisanej logiki oraz komunikacji (2.2.3) i wgranie ich na Arduino,
- przetestowanie poprawności działania algorytmów,
- zbudowanie trwałych nakładek na platformy (2.1.5),
- dobranie przekaźnika sterującego silnikiem (2.1.7),
- sprawdzenie, wraz z opiekunem pracy doktorem Markiem Długoszem, ilości wydzielanego ciepła przez przekaźnik pod obciążeniem (2.1.8).

Jak można odczytać z tabeli 2.3, wszystkie wymagania zostały spełnione, a więc projekt zakończył się sukcesem.

Pomimo konieczności spełnienia dokładnie jednej funkcji (szybkiego wyłączenia samochodu w momencie zagrożenia), istnieją pewne możliwości dalszego rozwoju projektu. Po pierwsze można rozszerzyć jego działanie do dodatkowego hamowania pojazdu, które jest osiągalne poprzez przełączenie silnika w tryb generatora. Po drugie, można wykorzystać komunikację między modułami A i B do prze-

syłania informacji diagnostycznych samochodu do operatora, co w dalej umożliwiłoby użycie projektu jako swoistego zdalnego centrum diagnostycznego samochodu.

Bibliografia

[1] J. M. Hughes. *Real World Instrumentation with Python*. O'Reilly Media, Inc., 2010. Chap. 8.

[2] S. Goyal. *Major Seminar On Feature Driven Development*. 2008. URL: *http://csis.pace.edu/~marchese/CS616/Agile/FDD/fdd.pdf*.

[3] *Arduino - ArduinoProBoard*. URL: *https://www.arduino.cc/en/Main/ArduinoBoardPro*.

[4] *Romoss - solo 5*. URL: *http://romoss.com/proview.php?ID=17*.

[5] *HC-12 Wireless Serial Port Communication Module, User Manual v1.18*. URL: *https://botland.com.pl/index.php?controller=attachment&id_attachment=1188*.

[6] Z. Wojcik. *3D Printed Case for Arduino Uno, Leonardo. by ZygmuntW*. 2015. URL: *http://www.thingiverse.com/thing:628929*.

[7] P. Banaszkiewicz. *Module cases*. 2015. URL: *https://github.com/pbanaszkiewicz/engineering-thesis/tree/master/cases/README.md*.

[8] *Fritzing*. URL: *http://fritzing.org/home/*.

[9] *Anly Electronics ASR-90DD-H datasheet*. URL: *http://www.anly.com.cn/english/pdf/ASR_KB.pdf*.

[10] *Przekaźniki półprzewodnikowe*. URL: *https://www.relpol.pl/content/download/37247/517211/file/techpol.pdf*.

[11] J. Bromirski. *Teoria automatów*. Wydawnictwa Naukowo-Techniczne, 1969. Rozd. 4.4.3.1. S. 181.

[12] *The Twofish Team's Final Comments on AES Selection*. 2000. URL: *http://www.schneier.com/paper-twofish-final.pdf*.

[13] *Arduino - Reference*. URL: *https://www.arduino.cc/en/Reference/HomePage*.

[14] P. Banaszkiewicz. *Engineering thesis*. URL: *https://github.com/pbanaszkiewicz/engineering-thesis*.

[15] *Konwerter USB-UART FTDI 3,3/5V*. URL: *http://botland.com.pl/konwertery-usb-uart-rs232-rs485/1611-konwerter-usb-uart-ft232-33-5v.html*.

Spis rysunków

Spis tablic