To już 9-ty odcinek kursu Raspberry Pi. Numery archiwalne MT z poprzednimi odcinkami można kupić na www.ulubionykiosk.pl

Zbudowanie własnego robota w oparciu o Raspberry Pi wcale nie jest trudne. Nie trzeba skończyć studiów inżynierskich. Wystarczy odrobina samozaparcia i kreatywności. Nie potrzebujecie też znacznych nakładów finansowych. Rozejrzyjcie się dobrze po szafach i okolicznych sklepach – nie tylko tych z elektroniką.

Raspberry Pi (9): mój pierwszy robot

Na rynku dostępnych jest wiele zestawów, które pozwalają budować roboty. Niestety, ich ceny zawierają się w przedziale od „drogie" do „niebotycznie drogie". Nowy, podstawowy zestaw Lego Mindstorms to wydatek grubo powyżej tysiąca złotych. A do poważniejszych projektów będziecie go musieli rozszerzyć. Oferty mniejszych firm czasami wydają się atrakcyjne. Jednak po otwarciu pudełka okazuje się, że zestaw wymaga wielu dodatkowych elementów, które znacznie podnoszą ostateczny koszt zabawy. Często są to też projekty robione pod specyficzne komponenty, które np. leżały na zapleczu. Bez poważnych przeróbek (np. otworów montażowych) nie da się ich zamienić na tańsze lub bardziej popularne. Dzieje się tak dlatego, że małe serie są niejednokrotnie opracowywane przy okazji np. prac naukowych czy zajęć hobbystycznych. Wtedy funkcjonalność nie zawsze idzie w parze z kalkulacją ekonomiczną. Jeżeli zdecydujecie się na zakup takiego zestawu, najpierw dokładnie sprawdźcie instrukcję, przestudiujcie listę elementów, których nie ma w zestawie i dowiedzcie się od producenta, jaki jest czas dostawy. Jeśli znaleźliście ciekawy projekt robota w Internecie, pamiętajcie też, że niektóre strony „wiszą" długo po zakończeniu działalności ich właścicieli.

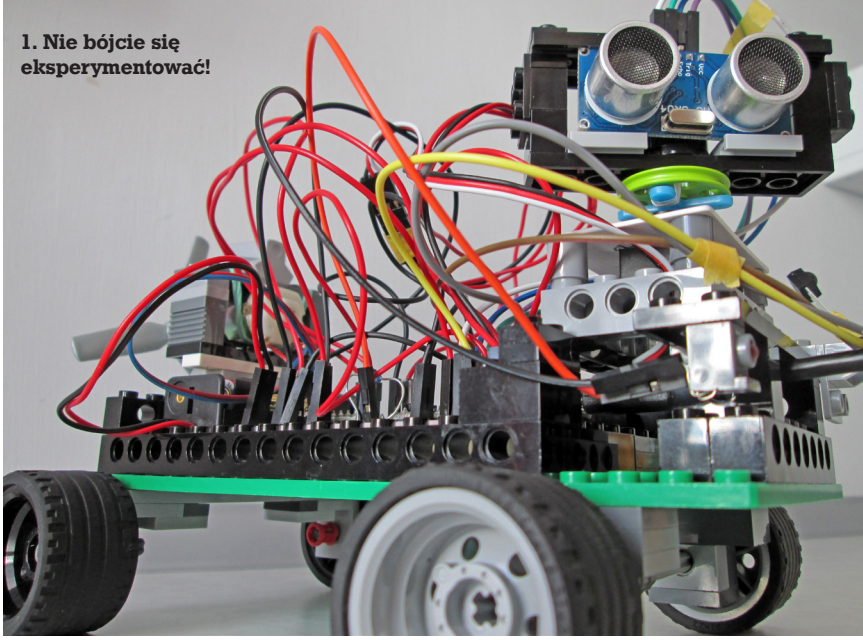
Druga kategoria to gotowe, złożone zabawki w stylu robo-dinozaur. Często są o wiele tańsze, ale nie można ich w żaden sposób modyfikować. Po pół dnia zabawy wracają więc do pudełka i nigdy stamtąd już nie wychodzą. W sklepach można też spotkać zestawy edukacyjne, typu „robot z puszki po napoju". Są ciekawe, szybkie do realizacji, zazwyczaj skupiają się na jednym aspekcie (np. wykrywanie ruchu, zasilanie słoneczne), a ich cena mieści się w budżecie prezentu urodzinowego dla kolegi. Niestety, szybko się nudzą.

Wbrew pozorom, **zbudowanie robota własnymi siłami wcale nie jest trudne**. Wymaga trochę wiedzy inżynierskiej, ale bardziej – kreatywnej wyobraźni. Przepisem na sukces jest poszukiwanie alternatyw dla gotowych rozwiązań. Przykładem niech będzie platforma robota, na której osadza się jego wszystkie mechanizmy. Można kupić gotową, np. Magician Chassis (ok. 70 zł). Jest estetyczna, kolorowa, zawiera wszystkie elementy – łącznie z dwoma silnikami, kołami i pojemnikiem na baterie – sprawdzi się doskonale. Jej złożenie zajmie kilka minut. Zastanawiam się jednak: czego tych kilka minut nauczy? Obsługi śrubokręta?

I tutaj dochodzę do *clue* budowy własnego robota. Jestem zdania, że w przypadku edukacji mechatronicznej, to **nie efekt końcowy jest najważniejszy, ale przebyta droga**, żeby go zrealizować. Oczywiście, cel musi być osiągnięty, zadanie skończone. To bardzo ważne, aby projekty kończyły się czymś realnym. Wydaje mi się jednak, że sam proces realizacji jest tutaj kluczowy. Chodzi o kształtowanie pewnego inżynierskiego podejścia do sprawy, patrzenia na problemy i poszukiwania dla nich rozwiązań (najlepiej zespołowego). Chodzi o to, żeby włączyć szare komórki, pokombinować, przemyśleć różne rozwiązania – ich wady i zalety. Zdobyte **doświadczenia na pewno zaprocentują** w przyszłości.

„Kup" kontra „zrób"

Zamiast kupować gotową platformę, może warto zrobić ją samemu? **Materiały znajdziecie wszędzie**. Większość hobbystów (zwłaszcza modelarzy) to typowe chomiki. Nawet wizyta w kawiarni może okazać się źródłem wielu ciekawych „surowców". Denka od kubka po kawie mogą posłużyć jako koła lub platforma startowa dla rakiety, grube słomki to idealne odbojniki dla łódki lub tunele do prowadzenia kabli. Ilość zastosowań drewnianych patyczków do mieszania kawy jest wręcz niezliczona. W projekcie, który zaraz Wam opiszę, do budowy platformy wykorzystałem tzw. **drewno cytrusowe**. Jak je zdobyć? Wystarczy pójść do najbliższego warzywniaka i poprosić o skrzynkę po owocach. Niektóre zrobione są z dykty – te są mniej przydatne – ale większość powstała z 3-milimetrowej, bardzo lekkiej sklejki. Za darmo (bo sprzedawcy je najczęściej po prostu wyrzucają) można uzyskać bardzo uniwersalny materiał. Projekt? Kartka, ołówek i Internet, żeby



1. Nie bójcie się eksperymentować!

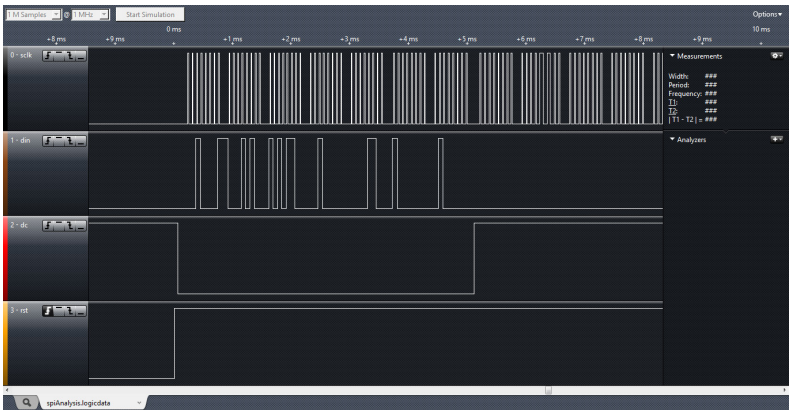
poszukać natchnienia i kart katalogowych komponentów. Oprogramowanie? W większości darmowe, np. Inkscape lub SketchUp do projektowania.

Oczywiście pewne elementy będziecie musieli po prostu kupić. Nikt przecież nie robi sam silników. Ale zdobyta wiedza i doświadczenie na pewno przydadzą się w przyszłości – nie tylko w dziedzinie inżynierii. Zadziwiające, jak dzieci potrafią przenosić doświadczenia między różnymi dyscyplinami. To pouczające, jakich zdolności szukają pracodawcy u kandydatów na pracowników.

Przygotowanie

Proponuję zacząć od **dobrej orientacji odnośnie sklepów we własnej miejscowości**. Zróbcie listę potrzebnych rzeczy. Niektóre materiały (np. śrubki o średnicy 2,5 mm do przymocowania Raspberry) można dostać jedynie w wyspecjalizowanych placówkach – są charakterystyczne dla konkretnej dziedziny i w markecie budowlanym ich nie znajdziecie. Różnice w cenach potrafią być bardzo duże. W jednym sklepie dystanse znalazłem po 40 gr/sztukę – w innym te same już po 1,20 zł. Niby kwota niewielka, ale wierzcie mi – szybko uzbiera się niezła sumka. Jeśli korzystacie ze sklepów internetowych, zwracajcie uwagę na koszty wysyłki. Dostawa drobnych elementów może okazać się dużo droższa niż marże sklepów stacjonarnych. I nie oszukujcie się. Zawsze czegoś zabraknie. Proponuję **drobne elementy kupować w pewnym nadmiarze**, na wypadek gdyby coś poszło nie tak. Znajdźcie jeden-dwa sklepy. Przy częstszych zakupach można liczyć na rabaty, punkty lojalnościowe albo bezproblemowo wymienić nietrafione elementy. W przypadku sklepów internetowych (aukcji) sprawdzajcie, czy czasem nie ma takich z siedzibą w Waszym mieście – są zazwyczaj tańsze. W zapasie miejcie jakiś sklep stacjonarny, który przyda się w sytuacji alarmowej.

Zwróćcie również uwagę na **wszelkie koszty dodatkowe**. W wielu poradnikach znalazłem stwierdzenia całkiem oczywiste dla ich autorów, np.: „przylutuj element do płytki”. Bardzo fajnie, przecież każdy ma w domu lutownicę, cynę, kilka kolorów kynaru (taki drucik do łączenia elementów na płytce) i odsysacz (gdy coś pójdzie nie tak). Normalka, leży w każdej szafie. Otóż niekoniecznie. I to właśnie takie koszty dodatkowe najczęściej dobijają początkujących. Proponuję więc dobrze przemyśleć całe przedsięwzięcie i **wcześniej zrobić listę brakujących narzędzi**. Popytajcie znajomych. Inżynierowie, modelarze, hobbyści – to bardzo otwarta i serdeczna brać. Można też przeanalizować swoje potrzeby pod kątem chudego portfela. Oczywiście miło mieć stację lutowniczą za 200 zł, większość swoich projektów zlutowałem jednak... 30-letnią kolbą pożyczoną od wujka (na tzw. wieczne nieoddanie). Z elegancką, ebonitową wtyczką. Jasne, że lepiej się pracuje z nowym sprzętem dobrych firm. Ale to kosztuje. Spróbujcie **zacząć coś robić z podstawowym zestawem narzędzi**, zamiast kompletować je przez kolejne trzy lata i dopiero wtedy myśleć o pierwszym projekcie.



2. Analiza przebiegów z pomocą oprogramowania Saleae Logic (dostępne za darmo na stronie producenta)

Najważniejsze **narzędzia na początek** to: mała piłka, pilniki, nóż z segmentowanym ostrzem, obcęgi, szczypce, kilka arkuszy papieru ściernego, linijka stalowa i kątomierz stalowy (żeby równo ciąć, unikajcie aluminiowych). Musicie mieć też **możliwość robienia otworów**. Może do tego służyć zwykła wiertarka, ale bardziej uniwersalna jest mała wiertarko-szlifierka (cena 80 zł i więcej, tanie potrafią być mało precyzyjne). Przy zabawie z elektroniką (oprócz wspomnianej lutownicy, kynaru), konieczny będzie chociaż najprostszy **miernik uniwersalny**. Podstawowe modele kosztują ok. 20 zł. Oprócz mierzenia napięć służą do upewnienia się, że połączenia zostały odpowiednio wykonane (wybierajcie te, które robią „bip!"). Do opcjonalnych narzędzi zaliczyłbym np. **zasilacz laboratoryjny**. Zasilacze takie dostarczają stabilne napięcie w szerokim zakresie (np. do 15 V) i o prądach 3 A oraz więcej. Podstawowe modele kosztują ok. 120 zł. Dzięki nim można testować układy, zanim użyjecie ostatecznego zasilania.

Chciałbym też zwrócić Waszą uwagę na **analizatory logiczne**. Są to urządzenia, które pozwalają sprawdzić, jakie sygnały cyfrowe podróżują między poszczególnymi komponentami. Daleko im do oscyloskopów, ale umożliwiają wstępną orientację, czy napisane przez Was procedury poprawnie generują odpowiednie sygnały. Na portalach aukcyjnych podobne urządzenia można spotkać już za 40 zł.

Jeżeli zdecydujecie się na zasilanie za pomocą pakietów LiPo, czeka Was także zakup odpowiedniej **ładowarki**. Niestety, nie da się jej niczym zastąpić – nawet nie próbujcie żadnych samoróbek. Uszkodzone LiPo eksploduje (dosłownie!) i pali się bardzo efektownie. Niektóre z ładowarek sprzedawane są w ekonomicznych wersjach bez zasilacza 220 V. Możecie się zdecydować na taki zakup, jeżeli jakiś leży już w Waszej szafie. Niestety, są one stosunkowo rzadkie – najczęściej wymagają 12-18 V przy dużych prądach, nawet 5 A. Przy zakupie ładowarki zwróćcie uwagę, czy jest wyposażona w **balanser**. To specjalny układ, który wyrównuje napięcie w poszczególnych celach baterii. To przydatna funkcja (acz niekonieczna), który przedłuży życie Waszych pakietów LiPo. Ceny ładowarek zaczynają się od 30 zł, ale dopiero te droższe zapewnią Wam możliwość rozładowywania pakietów, ładowania na potrzeby składowania itp.

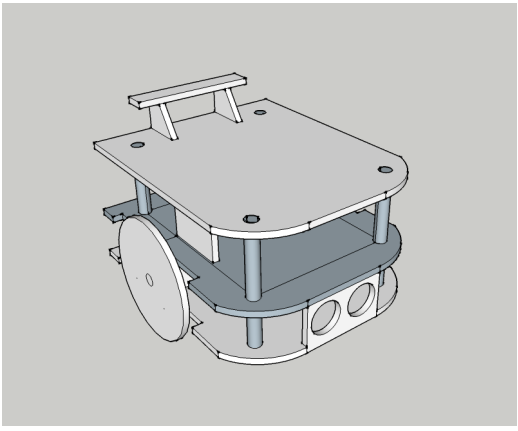
Jeszcze jedna uwaga: **miejsce pracy**. Upewnijcie się, że będziecie mieli gdzie tworzyć. Niewielu z Was (a właściwie: nas) stać zapewne na luksus posiadania własnego warsztatu. Większość korzysta z biurek, stołów kuchennych itp. Pamiętajcie o **dobrym oświetleniu** miejsca pracy i jego ochronie. Będziecie wykorzystywać różne narzędzia, więc lepiej żeby ślady ich używania nie pozostały na meblach. Polecam stosowanie jako ochrony np. resztek wykładziny podłogowej lub obrusów gumowanych. Sam mam taki – w kwiatki, widać go na wielu relacjach. Świetnie chroni stół przed rozlanymi płynami. Po skończonej robocie łatwo z niego zebrać i zetrzeć pozostałości. Podkładajcie maty samogojące, resztki płyt meblowych, nawet deski do krojenia – cokolwiek, co ochroni domowe sprzęty.

Zaawansowanym polecam również przemyślenie kwestii związanych z **ochroną przed wyładowaniami elektrostatycznymi** (ESD). Uziemiona mata, opaska na rękę powinny wystarczyć.

Planowanie

Zanim rzucicie się w wir pracy, warto jest usiąść i **przemyśleć cały projekt**. Dobry plan uchroni Was nie tylko przed stratą czasu, ale i niepotrzebnymi wydatkami. Na początku pojawia się oczywiście **pomysł**. Powinniście go ocenić pod wieloma względami (pamiętacie kryteria SMART, o których pisałem w nr 10/2014 MT?). Gdy Wasza analiza wypadnie pomyślnie i wiecie dokładnie, co chcecie zrobić – możecie zabrać się... za **dalsze planowanie**. Ja szkicuję. Tworzę bardzo dużo rysunków (o różnym stopniu szczegółowości), które pozwalają mi wyobrazić sobie rozmaite rozwiązania. I chodzi nie tylko o wygląd czy rozmieszczenie podstawowych elementów. Planuję, którędy będą przebiegać kable, punkty podporu, łączniki itp. Używam edytorów tekstu, programów graficznych, programów do mapowania myśli (ang. *mind mapping*) – przyznaję jednak, że najlepiej czuję się z zeszytem i ołówkiem. Polecam również – wspomniany już wyżej – **program do modelowania** 3D SketchUp firmy Tribal. Jego darmową wersję „Make" możecie pobrać ze strony www.sketchup.com (wersja „Pro" trochę kosztuje, ale Wam nie będzie potrzebna). Taki z grubsza sporządzony szkic 3D jest bardzo przydany. Zazwyczaj umieszczam na nim zarysy platformy, silników itp. Można sobie obracać planowany model i oglądać go pod różnymi kątami. Dzięki temu nieraz uniknąłem poważniejszych wpadek.

Oprócz samego pomysłu, celu do którego chcecie dążyć, warto **uświadomić sobie również ograniczenia**. W poprzednich tekstach wspominałem o czasie,



3. Szkic robota PiBotta w SketchUp

zakresie i środkach. W tego typu projektach powinniście również przewidzieć potencjalne zmiany, jakie pewnie wprowadzicie. Wasze roboty będą **wielokrotnie zmieniane i przebudowywane**. Będziecie podmieniać elementy, dodawać je lub usuwać, w zależności od potrzeb (np. czujniki). Co więcej – niektóre elementy (zazwyczaj te najdroższe) będą „wędrować" między różnymi projektami. Nie ma sensu ich za każdym razem kupować. Powinniście więc od razu myśleć o rozwiązaniach np. łatwo rozbieralnych. Ograniczcie do minimum liczbę elementów montowanych na stałe. Używajcie śrub i pasków zaciskowych zamiast kleju. Przykładem może być montaż pakietu zasilania. Jeżeli zdecydujecie się na koszyczek na akumulatory AA/AAA – teoretycznie można go przykleić na stałe (można użyć taśmy klejącej dwustronnej – kosztuje jedynie kilka złotych). Ale w przypadku migracji do pakietów LiPo trzeba będzie go oderwać. Może więc lepiej zamontować zasilanie np. za pomocą rzepów. Są tanie i jednocześnie zapewniają szybki demontaż. Oczywiście zawsze musicie myśleć o stabilności i sztywności konstrukcji.

Polecam też **używanie powtarzalnych i podobnych elementów** – np. śrubek o takim samym rozmiarze. Znacznie ułatwi to montaż i późniejsze serwisowanie.

Śledztwo

Kluczowym elementem zabawy jest jednak **zbieranie informacji**. Internet i prasa fachowa to nieocenione źródło. Czytajcie, czytajcie i jeszcze raz czytajcie. Jest bardzo duża szansa, że ktoś miał już podobny problem do tego, na jaki natrafiliście. Jeżeli znajdziecie jedno rozwiązanie, poszukajcie następnego i jeszcze jednego. Niestety – Sieć jest pełna informacji nie do końca wiarygodnych, niesprawdzonych lub po prostu starych. Jedynym sposobem na odsianie plewów, jest **porównywanie wielu źródeł.** Zwracajcie szczególną uwagę na datę opublikowania danej informacji. Sieć jest pamiętliwa – a robotyka przeszła w ostatnich czasach niezłą rewolucję. Niektóre rozwiązania mogą być po prostu nieaktualne.

Ja wyniki takiego śledztwa zapisuję lokalnie (na dysku), dbając o dodanie informacji o źródle, z którego pochodzą (np. adresu URL strony). Uchroni to Was przed zjawiskiem, które nazwijmy „dzwoni w którymś kościele". **Katalogowanie danych** to dodatkowy wysiłek, ale opłacalny. Zbieram wszystko – zwłaszcza noty katalogowe, schematy podobnych układów, zdjęcia gotowych produktów.

Śledźcie **specjalistyczne fora internetowe** i pytajcie internautów. O ile nie dublujecie tematów i zadajecie konkretne pytania – na pewno otrzymacie pomoc. Dodam, że „jak zbudować robota" pasuje raczej do wyszukiwarki niż na forum – tam Wasze pytania muszą być bardziej szczegółowe.

Nie bójcie się też **pytać o różne rzeczy w sklepach hobbystycznych**. Najczęściej pracują tam tacy sami pasjonaci jak Wy – a dodatkowo mający rozeznanie w rynku. Oczywiście, ich zadaniem jest sprzedaż produktów. Ale mądrzy sprzedawcy wiedzą, że warto uczciwie doradzać, bo lepiej mieć jednego stałego klienta niż dziesięciu jednorazowych. Na szczęście podobne poglądy są coraz bardziej popularne i – nawet jeżeli nie dokonacie żadnego zakupu – raczej nie spotkacie się z nieprzychylnością. W przeciwnym razie proponuję spacerek do innego sklepu. W mniejszych miejscowościach wybór może być ograniczony, ale jeżeli klientów zacznie ubywać, możecie mi wierzyć: jakość obsługi szybko się poprawi.

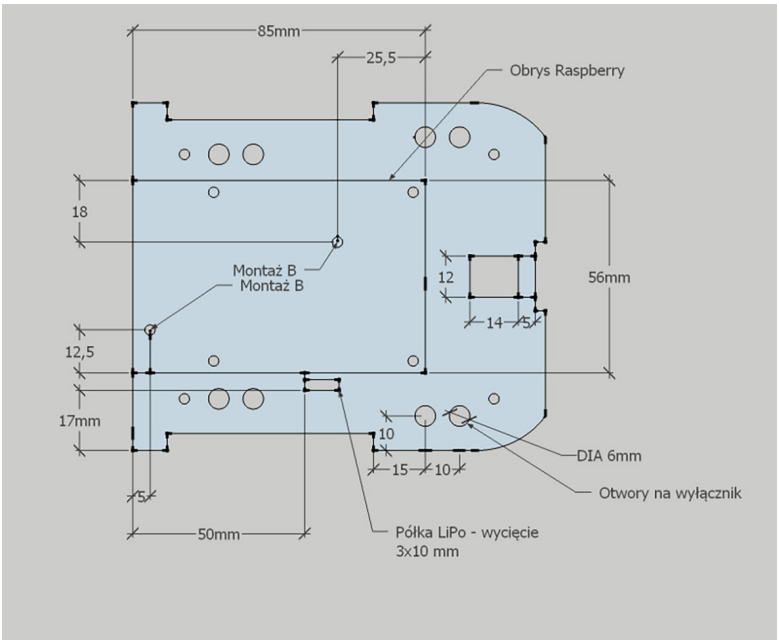
Programowanie

Dzisiejsze projekty mechatroniczne to nie tylko sam sprzęt (ang. *hardware*). Wiele zależy także od **oprogramowania** (ang. *software*). W przypadku typowych mikrokontrolerów, jak Arduino, wybór jest zazwyczaj ograniczony do języka C/C++. Raspberry, napędzany przez Linuksa, oferuje w tym zakresie znacznie większe możliwości. Możemy tworzyć w C/C++, ale i w Pythonie, Javie czy Scratchu i wielu, wielu innych (mniej lub bardziej egzotycznych). Do celów edukacyjnych preferowany jest zwłaszcza **Scratch**. Programowanie w nim **przypomina składanie klocków** (czy puzzli), odpowiadających kolejnym krokom programu. Wbrew pierwszemu wrażeniu, pozwala na całkiem dużo, a nie tylko na sterowanie uśmiechniętym kotem (postać z programu, coś jak żółw w Logo). Programy „pisze" się z użyciem **prostego interfejsu użytkownika** metodą „przeciągnij i upuść". Scratch jest **domyślnie instalowany na Raspberry**, razem z Raspbianem (najpopularniejszym systemem operacyjnym dla Raspberry). Ale jest również dostępny na innych platformach, w tym Windows. Polecam używanie wersji angielskiej. Inne języki programowania opierają się na poleceniach tylko i wyłącznie po angielsku. Używając angielskiego Scratcha, przyzwyczajacie się więc do instrukcji, które występują w Pythonie czy C. Przejście na nowy język będzie wtedy dużo łatwiejsze. Tych poleceń jest naprawdę niewiele. Nie trzeba ich wcale „wkuwać" na pamięć. Same wejdą do głowy podczas pisania kolejnych skryptów.

Mimo wszystko polecam jednak **Pythona**. Znajdziecie do niego bogaty zestaw bibliotek, które umożliwią realizację wszelkich zadań – od sterowania GPIO po komunikację po WiFi.

Zanim zdecydujecie się na jakiś konkretny język oprogramowania, upewnijcie się, że **pasuje do Waszych możliwości i celu**, jaki zamierzacie osiągnąć. Zakładając, że nie jesteście programistami (hobbystami czy zawodowcami), zrozumienie podstaw zajmie Wam trochę czasu. Im bardziej skomplikowane operacje Wasz robot ma wykonywać, tym trudniejszy będzie jego program.

Od czego zacząć naukę? Oczywiście od **poradników w Sieci**. Jest ich całe mnóstwo i na poziomie podstawowym naprawdę nie musicie kupować



4. SketchUp możecie również użyć do zaprojektowania wykroju dla plottera tnącego. Niestety, darmowa wersja umożliwia kreślenie tylko z dokładnością do 1 mm

żadnej książki. Pierwszą umiejętnością (oprócz podstaw składni), jaką musicie opanować, jest **właściwe czytanie i analizowanie czyichś programów**. To najlepszy i niezawodny sposób nauki. Pozwala wyrabiać dobre nawyki, korzystać z doświadczeń innych. W tym celu szukajcie pewnych źródeł kodu, gdzie dzielący się nim, dbają o jego czystość i poprawność. Z programowaniem jest jak w tym starym porzekadle – „czym skorupka za młodu...". Jeżeli od samego początku nauczycie się **pisać porządny kod**, zaoszczędzicie wiele czasu na śledzenie dziwnych problemów. Budujcie swoje programy na solidnych podstawach, modularnie, komentujcie je, trzymajcie się standardów, świadomie zarządzajcie pamięcią, wydajnością i typami zmiennych – a wszystko pójdzie gładko.

Testowanie

Preferuję **testowanie na każdym etapie** budowania urządzenia. Element, który zamierzam dodać do rozwiązania, staram się najpierw sprawdzić „na boku". Używam do tego np. płytki stykowej, zworek, zewnętrznego zasilania. W ten sposób upewniam się, że kolejne komponenty działają. Czasami może to wymagać zbudowania czegoś naprawdę tymczasowego, łączonego za pomocą gumki-recepturki i taśmy klejącej. Ale zazwyczaj takie działanie się opłaca i pozwala na znalezienie problemów na bardzo wczesnym etapie. Inżynierowie nazywają taką procedurę tworzeniem ***proof of concept*** **(PoC)**. Chodzi o sprawdzenie (udowodnienie), czy dane rozwiązanie ma sens, jest możliwe do zbudowania i czy po prostu działa (oczywiście w pewnym przybliżeniu).

Dotyczy to również kodu źródłowego oprogramowania sterującego. Często tworzę **wiele wersji kodu**, zanim uzyska on ostateczny kształt. Do szybkiego prototypowania używam Pythona. Jeżeli zaczyna mi zależeć na wydajności lub efektywności – przerzucam się na C/C++. Każdy z etapów projektu może mieć różne wymagania i wybrane narzędzia powinny być do niego adekwatne.

Mój robot czyli PiBotta

Nie ma lepszego sposobu na naukę, jak **realizacja swoich pomysłów**. Postawienie sobie konkretnego celu gwarantuje nie tylko przyswajanie praktycznej wiedzy, ale i satysfakcję z jej materialnych efektów.

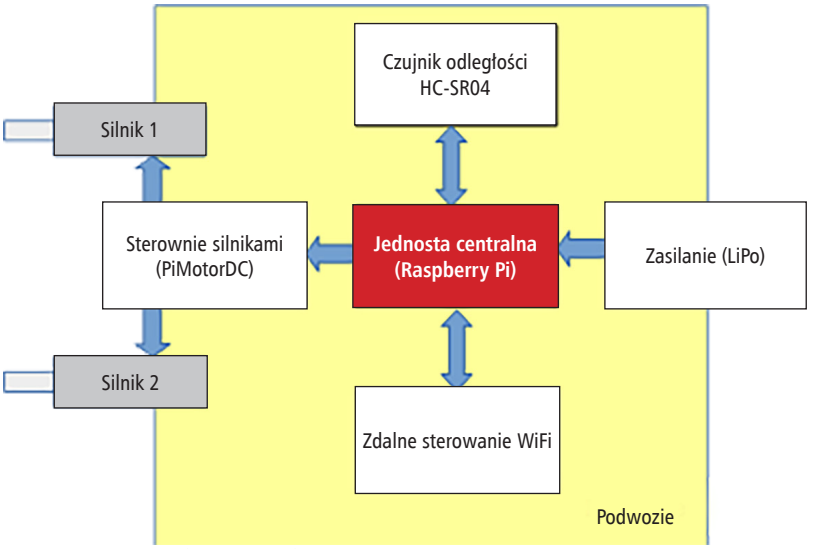
Czy może być coś bardziej ekscytującego od własnego robota? Zdecydowałem się więc na **zbudowanie robota mobilnego** w klasycznym układzie 3-punktowym. Dwa koła umieszczone po bokach napędzane są przez silniki (niezależnie). Trzeci punkt podparcia to obrotowa kulka. Na źródło zasilania wybrałem pakiet LiPo 7,4 V. Czujnik odległości obserwuje otoczenie. Komunikację ze światem zewnętrznym zapewnia moduł WiFi. Całość jest kontrolowana przez Raspberry Pi oraz rozszerzenie PiMotorDC (msx-elektronika.pl) do sterowania silnikami.

Celem mojego projektu było **zbudowanie podstawowej platformy**, która w przyszłości ma posłużyć do dalszych eksperymentów.

Projektowanie i budowa takiego robota składa się z bardzo wiele szczegółów, detali i kompromisów. Ich dokładne opisanie zajęłoby wiele stron. Poniżej skoncentruję się na głównych koncepcjach. Szczegóły – łącznie z rysunkami technicznymi i montażowymi – znajdziecie na stronie projektu www.uczymy.edu.pl/moodle. Szukajcie kursu „PiBotta: zbudujmy go razem" (jest dostępny za darmo, po zalogowaniu jako „gość" [2]).

Jednostka centralna

Sercem każdego robota jest **jednostka centralna**. To ona wykonuje program, kontroluje pracę



5. Ogólna koncepcja robota PiBotta

całego urządzenia, podejmuje decyzje na podstawie danych zbieranych z otoczenia lub instrukcji otrzymywanych od operatora. Oczywiście możecie sobie zadać pytanie, czy Raspberry jest najlepszym z możliwych wyborów. To zależy od Waszych planów. Jeżeli zamierzacie ograniczyć się do podstawowych funkcji i czujników – pewnie wystarczy Arduino. Ale jeżeli myślicie np. o podłączeniu kamery i efekciarskiego wyświetlacza TFT – nie ma to jak Linuks pod maską!

Jednym z ostatnich modeli przedstawionych przez Fundację Raspberry jest **B+**. Wyróżnia się mniejszym zapotrzebowaniem na prąd i dodatkowymi portami GPIO (40 pinów zamiast 26) – co zwiększa możliwości w zakresie sterowania różnymi elementami elektronicznymi. Kolejnym jego atutem jest bardziej zwarta budowa (np. niższy profil i karta microSD) i cztery regularnie rozmieszczone otwory montażowe, umożliwiające stabilne przymocowanie Raspberry do szkieletu robota. No i oczywiście jest jeszcze wersja A+ – o ponad 2 cm krótsza od standardowych B i z jeszcze mniejszym apetytem na ampery. Całkowita „świeżynka", czyli wersja Pi 2 z czterordzeniowym procesorem Cortex A7, zapewne akurat w tej dziedzinie nie wniesie zbyt wiele.

Raspberry oferuje zdecydowanie **więcej możliwości** niż tradycyjnie używane do takich zastosowań mikrokontrolery. Wiele rzeczy ułatwia, choć pod pewnymi względami jest znacznie bardziej wymagająca (zasilanie!). Lubię traktować Raspberry jako **narzędzie do prototypowania**. Dopiero gdy naprawdę i dokładnie wiem, w którą stronę pójdzie projekt, jeżeli to możliwe – zastępuję go czymś bardziej zwartym i – co tu dużo mówić – tańszym. Chyba, że po prostu chcę poeksperymentować. Wtedy nie ma lepszego narzędzia!

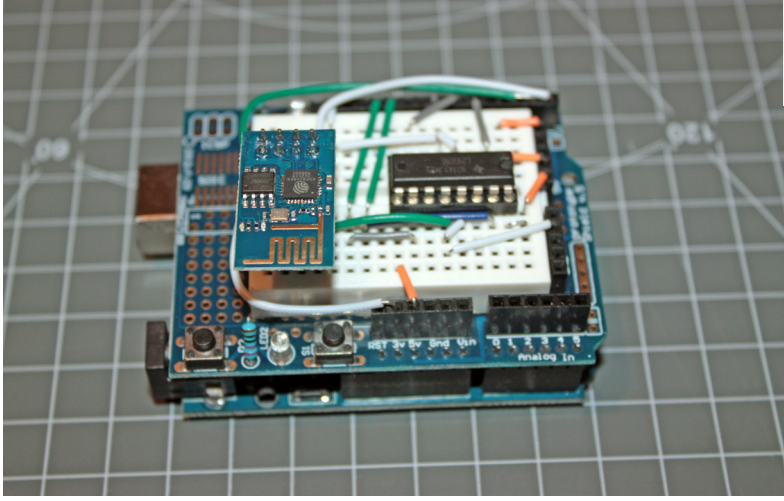
Zdalne sterowanie – komunikacja z robotem

Komunikacja z robotem to kolejne wyzwanie dla każdego inżyniera. Taki kanał służy nie tylko do **wydawania poleceń**, ale i **odbierania danych**, np. transmisji z pokładowej kamery wideo. W przypadku Raspberry wybór jest właściwie oczywisty – karta WiFi włożona do portu USB. Taka bezprzewodowa transmisja zapewnia wystarczające parametry przepustowości. Co więcej, dzięki Linuksowi jest łatwa w konfiguracji. W ten sposób możecie **kontrolować swojego robota** nie tylko z laptopa, ale z dowolnego smartfonu lub innego urządzenia wyposażonego w WiFi.

W moim projekcie łączyłem się do robota za pomocą SSH. Na Raspberry uruchamiałem prosty skrypt Python'owy, który przejmował sterowanie nad napędami i zbieraniem informacji z czujników. Nic jednak nie stoi na przeszkodzie, żebyście uruchomili na RPi serwer WWW, funkcję punktu dostępowego (ang. *access point, AP*) i sterowali nią z poziomu przeglądarki.

Jeżeli chodzi o same **karty WiFi**, na rynku znajdziecie ich bardzo wiele. Tak naprawdę od producenta ważniejszy jest **układ realizujący transmisję** (znajdujący się w ich wnętrzu). Popularnymi kartami są Realtek'i, np. serii 8192. Wszystko sprowadza się do pytania nie „czy" – ale **„ile" zajmie Wam uruchomienie zakupionej przez Was karty**. Jeżeli traficie na chipset, do którego wsparcie jest już w jądrze Linuksa – taka karta wystartuje sama i natychmiast. Czasami jednak będziecie musieli dodać odpowiednie sterowniki (pliki *.ko) i firmware (*.bin). Ja użyłem karty z RTL8188CU, zgodnie z instrukcjami znalezionymi na stronie [1]. Zauważcie, że są one uzupełniane dla każdej kolejnej wersji Raspbiana. Generalnie społeczność fanów Raspberry jest w tym temacie bardzo aktywna i nie powinniście mieć żadnych kłopotów ze znalezieniem rozwiązania dla kupionej przez Was karty.

Do niedawna technologia WiFi zarezerwowana była tylko dla bardziej wydajnych jednostek. Cóż, sytuacja w mechatronice zmienia się jak w kalejdoskopie. Jedną z nowinek jest układ ESP8266, dostępny w kilkunastu różnych wersjach (np. ESP-01). Za ok. 3$ Wasze Arduino może posmakować „Internet of Things".



6. Trochę inne podejście do sterowania robotem – „mózg” ArBotta: Arduino UNO, płytka prototypowa, sterownik L293 i WiFi z użyciem ESP8266

Czujniki

Czujniki to zmysły robota. Pozwalają mu **obserwować otoczenie** i na podstawie uzyskanych w ten sposób danych **dostosowywać swoje zachowanie** do warunków. Na rynku znajdziecie bardzo dużo różnych rodzajów czujników. Mogą to być **sensory odległości** (np. dla robotów typu sumo), **ruchu** (np. czujki PIR robotów dozorujących), **koloru** (np. dla robotów typu *line-follower*), **kontaktowe**, aż po **kamery termowizyjne**. Możliwości są praktycznie nieograniczone. Co więcej – większość dostaniecie w formie kompletnych modułów. Wystarczy podłączyć je kilkoma kabelkami do GPIO, dodać parę linijek kodu w Python'ie i Wasz robot stanie się świadomy piękna otaczającego go świata.

W swoim robocie użyłem prostego **czujnika odległości HC-SR04**. Pozwala on na określenie dystansu do przeszkody w efektywnym zakresie 4 do 500 cm (katalogowo; w rzeczywistości stabilny zasięg jest dużo mniejszy, do 2 metrów). Jeden taki czujnik wystarczył, żeby skręcać przed szafkami lub unikać taranowania pudełek (zobaczcie tutaj: **www.youtube.com/watch?v=B9F-4TOfv3Y**). Niestety, w przypadku węższych przedmiotów będziecie musieli wymyślić coś bardziej finezyjnego. I pewnie droższego.

Chciałbym Wam tu zwrócić uwagę na kwestie związane z **napięciem logiki**. Jak zapewne pamiętacie, Raspberry posługuje się logiką 3,3 V. Napięcie 3,3 V oznacza „1”. Niektóre z czujników – jak HC-SR04 – preferują logikę 5 V. Ponieważ chodzi raczej o pewien zakres niż konkretną wartość, czujniki zrozumieją pobudzenie (wejście) z Raspberry na poziomie 3,3 V. Ale odpowiedź 5 V może skutecznie uszkodzić RPi. Żeby się przed tym ustrzec, stosuje się **konwertery poziomów logicznych**. W bezpieczny sposób pośredniczą one w wymianie sygnałów między urządzeniami o różnych napięciach logiki. Inne rozwiązanie to **dzielniki napięcia** oparte na rezystorach.

Kolejnym wyzwaniem mogą być **czujniki zwracające sygnał analogowy**. Niestety, Raspberry nie wyposażono w przetwornik analogowo-cyfrowy (ADC). Ale Arduino tak! Może warto połączyć ich siły?

Zasilanie

Zasilanie to jeden z krytycznych punktów zabawy. Praktycznie sprowadza się do dwóch parametrów: **napięcia i prądu**. Silniki potrzebują zazwyczaj 3-12 V, Raspberry: 5 V, czujniki 3,3 V lub 5 V. Każdy element robota będzie wymagał też odpowiedniego prądu. Sam Pi wymaga co najmniej 350 mA, ale niektóre napędy nawet 2 A! Wybrane źródło zasilania musi dostarczyć odpowiednie napięcie i prąd (czasem potrzebne są odpowiednie układy dopasowujące).

Możliwości rozwiązania tego problemu ograniczają się właściwie do wyboru między akumulatorkami NiMH (popularny rozmiar AA lub AAA) lub pakietem LiPo (ew. LiFe). Ze względu na znaczny ciężar akumulatorków NiMH (potrzeba ich 6xAA) oraz ograniczone miejsce, zdecydowałem się na **pakiet LiPo**. Taki wybór związany był z dodatkowymi kosztami nie tylko w postaci odpowiedniej ładowarki, ale i miernika poziomu napięcia baterii (który ostatecznie zamontowałem na boku robota). Napięcie pakietu trzeba kontrolować, gdyż zbytnie rozładowanie pakietów może doprowadzić do ich uszkodzenia.

W jednym z poprzednich artykułów opisywałem ciężarówkę zasilaną przez baterię „alarmową” do telefonu. To też może być jakieś rozwiązanie – choć wyjścia USB są dość niezręczne, gdy będziecie próbowali się do nich podłączyć na ograniczonej przestrzeni (kątowe wtyczki USB niełatwo dostać i są dość drogie).

Zwróćcie również uwagę na **kable**. Nie przez przypadek te od pakietu są tak grube. Użyty przeze mnie **Redox 2S 1300 mAh 20 C** może dostarczyć ponad 25 A (20*1,3) prądu! Na wszelki wypadek dobierajcie kable **wytrzymujące przynajmniej 20% więcej prądu** niż maksymalne spodziewane natężenie (np. przy zablokowanych wałach silników).

Zawsze pamiętajcie o zapewnieniu sobie **przerwania zasilania baterii za pomocą wyłącznika** – tak ze względów praktycznych, jak i bezpieczeństwa. Wtyczki od pakietów są tak wykonane, żeby łatwo się nie rozłączyły (różne rzeczy dzieją się z modelami). Podobnie jest z Dean (T) Redox'a. Startowanie robota poprzez połączenie do obwodu to kiepski pomysł. A wyłączanie robota przez rozłączanie – jeszcze gorszy.

Napęd

Wybór napędów do robotów jest stosunkowo szeroki. W tym celu możecie użyć silników prądu stałego (DC), silników krokowych lub serw modelarskich.

Tabela 1. Różne rodzaje napędów

Napęd	Zalety	Wady	Zastosowanie
Silniki krokowe	Możliwość precyzyjnego sterowania. Można je pozyskać z np. starych drukarek.	Wymagają większych prądów; skomplikowane sterowanie (duża ilość wyprowadzeń do obsłużenia), są stosunkowo wolne, ale przede wszystkim duże i ciężkie.	Większe i cięższe roboty mobilne, roboty stacjonarne, drukarki 3D.
Serwomechanizmy	Bardzo popularne i łatwo dostępne; łatwo się je steruje (PWM, 1 kabel sygnałowy), charakteryzują się dużą precyzją i ograniczonym apetytem na prąd. Szeroki wybór - od małych i lekkich (nawet 2-3 gramy!) aż po duże i bardzo wydajne. Niska cena.	Zwykłe serwomechanizmy obracają się jedynie o 180°; można je przerobić, w przypadku niektórych nietrywialne (+utrata gwarancji). Dostępne są również serwa 360, ale ich cena zazwyczaj mocno przekracza 40 zł/sztukę.	Uniwersalne – modelarstwo, robotyka.
Silniki DC (mój wybór)	**Duża moc i szybkość obrotowa; niska cena.**	**Wymagają dodatkowych przekładni, które ograniczą obroty i zwiększą moment obrotowy. Są mało precyzyjne.**	**Roboty mobilne.**

Silniki DC są najszybsze – pozwalają na uzyskanie całkiem sporych obrotów. Do tego stopnia, że nie można ich zastosować bezpośrednio, tylko poprzez odpowiednią przekładnię. Zredukuje ona nominalne naście-tysięcy obrotów do np. rozsądnych 150 obrotów na minutę (przy 12 V). Przekładnia jest najczęściej zespolona z silnikiem na stałe, a katalogi podają ilość obrotów na przekładni (nie samego silnika). Możecie również kupić bardzo ciekawe przekładnie Tamiya, które składa się samodzielnie z zestawu elementów. Wtedy sami zdecydujecie o ich przełożeniach.

Sterowanie silnikami DC wymaga zewnętrznego zasilania i dodatkowych układów kontrolujących. Nie można takiego silnika podłączyć bezpośrednio do pinów GPIO. Nawet na biegu jałowym pobierają one często co najmniej 70 mA prądu, co już przekracza możliwości portów Raspberry (ok. 20 mA). Pod obciążeniem (np. gdy toczą całego robota) prąd ten znacznie rośnie, dochodząc nawet do np. 1,8 A przy zablokowaniu wału! Co więcej, sama ilość zakłóceń, które takie silniki wprowadzają w obwodzie, mogłaby być niebezpieczna nie tylko dla bezpośrednio podłączonego RPi, ale większości kontrolerów. Dlatego też do obsługi silników DC stosuje się dodatkowe układy. Dzięki nim za pomocą niskonapięciowej logiki można przełączać duże prądy zasilania. Popularnym układem realizującym takie zadania jest mostek L293 (i jego wariacje).

Niestety, sterowanie silnikami DC jest mało precyzyjne. Nigdy do końca nie wiadomo, kiedy się zatrzymają i ile obrotów zrobią w ciągu sekundy. Można oczywiście wyposażyć je w dodatkowe enkodery, ale wtedy ich cena rośnie szaleńczo.

Trochę inaczej działają **silniki krokowe**. Ich konstrukcja umożliwia bardzo dokładne pozycjonowanie, tzn. można kontrolować, o ile obróci się wał z dokładnością co do stopni czy ich części. Ich prędkość obrotowa nie zależy od napięcia czy prądu – ale od ilości impulsów sterujących. Kolejną zaletą jest stosunkowo duży moment obrotowy przy małych prędkościach. Ich wady to niewielkie prędkości obrotowe, rozmiary, waga oraz dużo bardziej skomplikowane sterowanie.

Kolejną możliwością napędu są **serwa modelarskie** – najłatwiejsze w sterowaniu i zasilaniu. Zazwyczaj pracują w okolicy 4-6 V – wystarczy więc zestaw czterech akumulatorów AA. Najmniejsze serwa udaje się zasilić z samego Raspberry, ale to rozwiązanie mające znaczenie raczej edukacyjne niż praktyczne (i niezalecane). Za to sterować można je bezpośrednio z Raspberry, podając sygnał PWM na kabel sterujący (zazwyczaj żółty lub biały). Ich ceny są często niższe niż większości silników DC czy krokowych. Problem z serwami polega jednak na tym, że oferują ograniczony zakres ruchu. Typowe serwa poruszają się +/-90° od pozycji neutralnej. W takiej konfiguracji nie nadają się oczywiście do poruszania kołem. Ale można je przerobić.

Przeróbka polega najczęściej na usunięciu (albo ominięciu) ich elektroniki, usunięciu niektórych trybów lub ich modyfikacji (np. przez ścięcie ząbków ograniczających ruch). Nie jest to zadanie specjalnie skomplikowane, ale wiąże się z pewnym ryzykiem zniszczenia serwa (i utratą gwarancji). Takim przerobionym serwem steruje się już oczywiście jak 5 V silnikiem DC – ale nadal zachowując znacznie większą precyzję. Oczywiście w sprzedaży znajdziecie firmowe serwa 360 – ale są rzadkie, a ich ceny nonsensowne.

Silniki DC

Do mojego projektu zdecydowałem się na **użycie silników DC**. Mają one tylko dwa wyprowadzenia. Sterowanie odbywa się poprzez zamianę biegunów (obroty lewo/prawo) i zmianę wypełnienia sygnału sterującego PWM (szybciej/wolniej).

Jeżeli chodzi o parametry silników, przede wszystkim należy zwrócić uwagę na **napięcie zasilania**. Jest to wartość charakterystyczna dla danego silnika. Zazwyczaj podaje się:

- napięcie nominalne (w woltach – [V]);

- zakres dopuszczalnych napięć – minimalne do maksymalne;

Napięcie nominalne określa optymalny punkt pracy silnika. Przykładowo, silnik DC o napięciu nominalnym 12 V może mieć **zakres dopuszczalny** 4,5-13,5 V. Przy napięciu mniejszym niż nominalne (12 V), będzie wolniej się obracać i szybciej zużywać. Poniżej napięcia minimalnego (tu: 4,5 V) przestanie się obracać. Zasilenie napięciem większym niż maksymalne (tu: 13,5 V) może uszkodzić silnik lub znacznie skrócić jego żywotność.

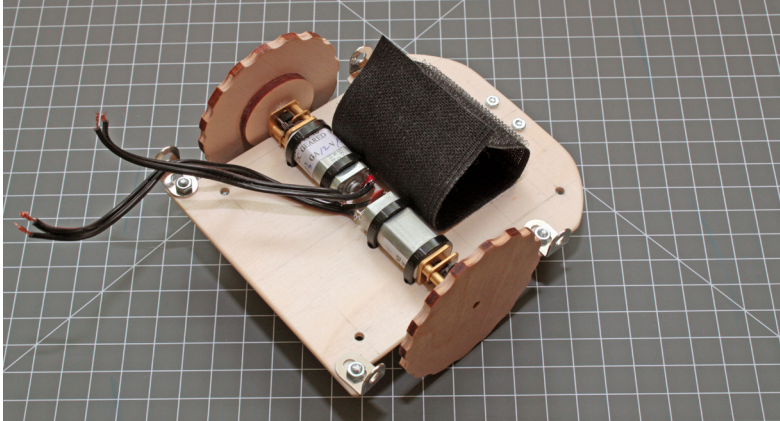
Kolejnym ważnym parametrem są **prądy pobierane przez silniki**:

- pobór prądu na biegu jałowym (w miliamperach – [mA]);
- pobór prądu w szczycie (w miliamperach – [mA] lub amperach [A]).

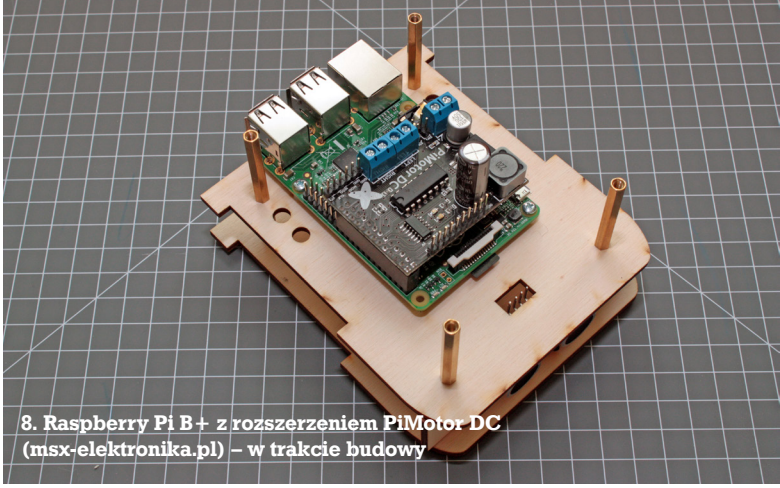
Silnik pracuje na biegu jałowym, gdy kręci się bez żadnego obciążenia. Silnik pobiera prąd szczytowy w przypadku zatrzymania jego wału. Obydwa prądy są bardzo ważne, gdyż **stawiają konkretne wymagania co do wydajności źródła zasilania i elementów sterujących takim silnikiem**. Teoretycznie źródło zasilania musi dać co najmniej prąd szczytowy. Chociaż powiem wam, że w praktyce sytuacja całkowitego zablokowania wału jest raczej rzadka. Nawet jeżeli robot centralnie wjedzie w ścianę, jego koła zaczną się po prostu ślizgać, a pobierany prąd będzie daleki od maksymalnego.

Sterowanie silnikami

Problem z silnikami polega na tym, że do pracy **potrzebują dużych prądów** – znacznie większych, niż jakikolwiek minikomputer lub mikrokontroler może sam wygenerować czy przełączać. Do sterowania konieczne są tu układy takie, jak wspomniany już L293 czy np. DRV8833 (wykorzystany wcześniej do silnika lego ciężarówki). W tym projekcie zdecydowałem się na użycie **gotowego rozszerzenia PiMotorDC** polskiej firmy msx-elektronika.pl. Produkt nakłada się bezpośrednio na złącze GPIO. Wystarczy podpiąć do niego zasilanie o napięciu odpowiednim do silników. Płytka ma na sobie przetwornicę 5 V, która zasila Raspberry poprzez pin 2 (5 V) – tzw. *back power*. Dzięki temu jedno źródło może być wykorzystane na potrzeby RPi i silników. To znacznie upraszcza cały projekt. Niewykorzystane porty GPIO wyprowadzone są na PiMotorDC, co umożliwia podłączenie dodatkowych peryferiów.



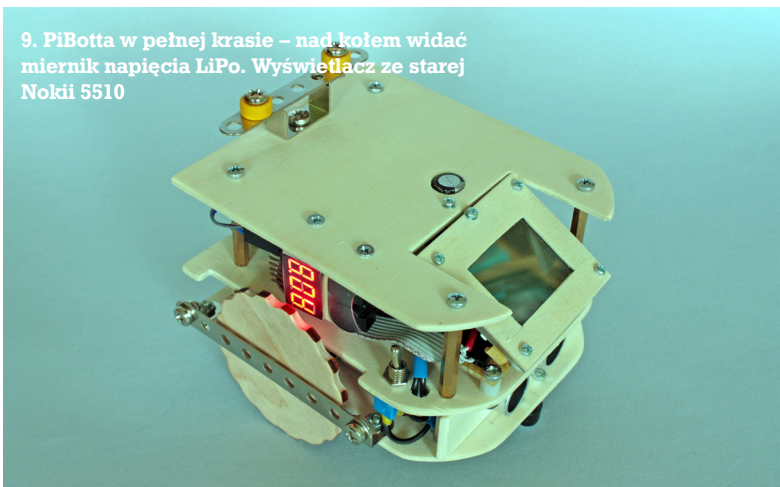
7. Dolna platforma robota mobilnego PiBotta z silnikami MT60 i rzepem na pakiet LiPo



8. Raspberry Pi B+ z rozszerzeniem PiMotor DC (msx-elektronika.pl) – w trakcie budowy

Podwozie i szkielet

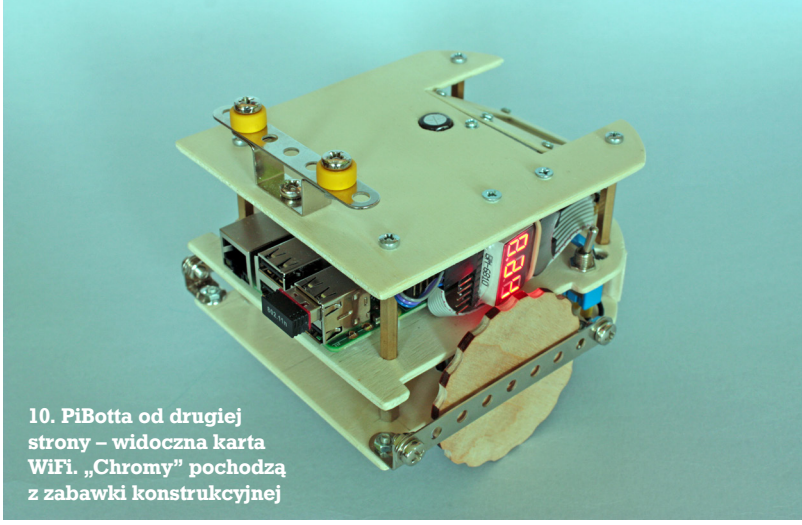
Wszystkie powyższe elementy muszą być **osadzone na wspólnym szkielecie**. Dla mojego robota zdecydowałem się na **trzypunktowe podwozie**. Przednie podparcie stanowi obrotowa kulka. Z tyłu znalazły się



9. PiBotta w pełnej krasie – nad kołem widać miernik napięcia LiPo. Wyświetlacz ze starej Nokii 5510

dwa niezależnie napędzane koła. Kolejne poziomy robota montowane są do poprzednich z użyciem metalowych dystansów. **Górna platforma** scala i stabilizuje całość.

Jest to **bardzo prosta konstrukcja**. Jej dodatkowe zalety to lekkość i łatwość dostępu do różnych komponentów (czyli serwisowania). Choć przyznaję, wygląda raczej... inżyniersko. Pewnie eleganckie opakowanie nadałoby robotowi PiBotta bardziej wyrafinowany wygląd. Dla mnie ważne były **praktyczność rozwiązania** i **możliwość łatwego rozbudowywania**.



10. PiBotta od drugiej strony – widoczna karta WiFi. „Chromy" pochodzą z zabawki konstrukcyjnej

Przy projektowaniu należy również zwrócić uwagę na **odpowiednie rozłożenie ciężaru**. Zbytnie przesunięcie ciężkich elementów do tyłu może sprawić, że niewielkie nierówności (np. podjazd pod krawędź dywanu) oderwą przednią kulkę od powierzchni, co w rezultacie może doprowadzić do dachowania.

Kolejnym ciekawym zagadnieniem jest **dobranie odpowiednie kół**. Jeżeli wybierzecie gładkie, robot będzie się ślizgał np. na płytkach podłogowych czy panelach. Przyczepność poprawi obciągnięcie kółek uszczelką do rur PCV lub gumą z dętki rowerowej. Najlepszym rozwiązaniem w przypadku wykładzin lub dywanów jest jednak płytkie ponacinanie kół na obwodzie. Jeżeli zrobicie wąskie nacięcia, robot nie będzie podskakiwał nawet na równej nawierzchni.

Oczywiście polecam Wam bawienie się kształtami i materiałami. Łączenie różnych technik w jednym obiekcie może dać czasami wspaniałe rezultaty. **Nie bójcie się eksperymentować** i próbować, opracowując własne rozwiązania.

Podsumowanie

Temat budowania robotów jest bardzo obszerny. W niniejszym artykule dotknąłem jedynie niektórych i najbardziej popularnych problemów z tego zakresu (podziękowania dla p. Pawła Dejnaka, który pomógł mi w realizacji części mechanicznej!). Mam jednak nadzieję, że odpowiedziałem na kilka z gnębiących Was pytań. Polecam konstrukcję takiego urządzenia – zwłaszcza zdalnie sterowanego i autonomicznego. Mogę Was zapewnić, że mimo znacznej ilości wyzwań, satysfakcja z ostatecznego rozwiązania wynagrodzi z nawiązką cały włożony wysiłek. A kolejny Wasz robot powstanie już znacznie szybciej i będzie bardziej zaawansowany.

Kto wie – a może trzeci wyląduje na Marsie? ■

Arkadiusz Merta

[1] http://www.mendrugox.net/2013/08/tp-link-tl-wn725n-v2-working-on-raspberry-raspbian/
[2] www.uczymy.edu.pl/moodle

ODPOWIEDZI NA PYTANIA

Dostaję od Was listy na temat popularnych problemów dotyczących pracy z Raspberry Pi. Poniżej kilka z pytań:

1. Potrafię już uruchomić Raspberry, teraz chciałbym wykorzystać go do nauki elektroniki. Czego potrzebuję?

Zanim chwycicie za lutownicę, do doświadczeń z elektroniką polecam zaopatrzyć się w różne narzędzia do prototypowania. Będzie to przede wszystkim płytka stykowa z zestawem zworek i kabelków połączeniowych. Proponuję kupować większe płytki, np. 830 pól. Do połączenia z Raspberry będziecie potrzebowali kabelków męsko- (wtykane do płytki) -żeńskich (gniazdo, do pinów GPIO Raspberry) lub akcesoriów w stylu opisywanego na łamach MT ProtoPi (http://www.mt.com.pl/raspberry-pi-kolejne-akcesoria i http://www.mt.com.pl/nowe-akcesoria-do-raspberry-pi-b). Do łączenia elementów na samej płytce potrzebne są kabelki męsko-męskie. Polecam używanie zworek. Komplet 140 sztuk kosztuje ok. 15 zł. Dzięki nim unikniecie gąszczu przewodów. Bardzo praktyczne są też zestawy zasilające płytki stykowe. Pozwalają na podłączenie zasilacza USB lub np. 12 V przez wtyk jack DC, zawierają stabilizatory do 3,3 lub 5 V i wyłącznik główny. Nie zapomnijcie też o multimetrze (nawet te tańsze wystarczą na początek) i proponowanym powyżej analizatorze stanów logicznych.

2. Szukam wtyku konektora T na płytkę stykową...

Konektor T (lub Dean) to rodzaj złącza używanego w pakietach LiPo firmy Redox. Pakiet ma gniazdo żeńskie, co chroni przed przypadkowym zwarciem. Lubię tę wtyczkę w zastosowaniach modelarskich. Złącze jest pewne, trzyma się mocno i trudno podłączyć je odwrotnie (nawet w ferworze zabawy). Niestety, nie spotkałem się z przejściówkami T na płytkę stykową lub przewlekaną. Zazwyczaj po prostu lutuję kable do wtyczki (męskiej) i dodatkowo chronię połączenia koszulkami termokurczliwymi. Ze względu na budowę wtyczki nie jest to wdzięczne zajęcie. Usługę taką oferują niektóre sklepy ze sprzętem RC. Mają doświadczenie i odpowiednie przewody, a kosztuje to najczęściej kilka zł (plus materiały).

3. Mój Raspberry Pi model B restartuje się po włożeniu karty bezprzewodowej do złącza USB. Czy jest to normalne?

Niestety tak. Przy podłączaniu urządzeń USB do działającego Raspberry (ang. *hotplug*), niektóre akcesoria mogą pobierać prądy przekraczające możliwości zasilania Pi. Powoduje to restart. Lepiej jest więc dołączać je przy wyłączonym Raspberry. Problem ten w większości przypadków nie istnieje w modelach serii A+/B+.

Szanowni Czytelnicy. Ewentualne pytania do autora można kierować bezpośrednio na adres: arkadiusz.merta@mt.com.pl