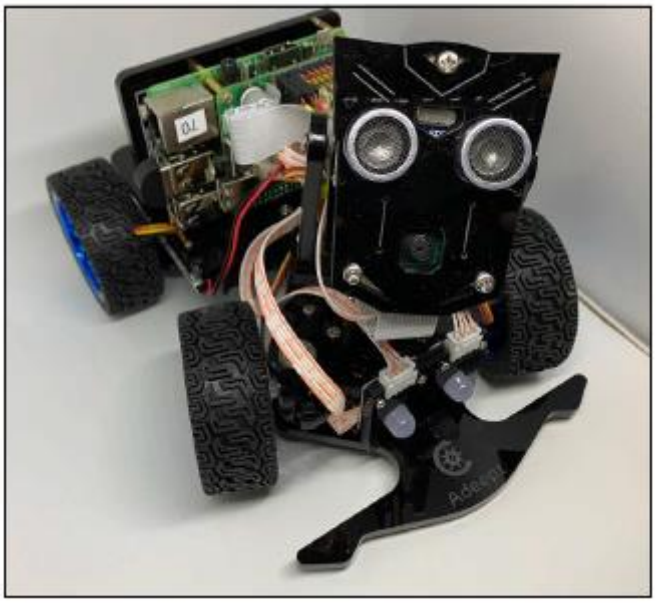
Budowanie pierwszego robota w Pythonie

W tej Części otwieramy robota, abyś zajrzał do środka, i pokazujemy, jak rozmawiać ze wszystkimi częściami za pomocą Pythona. Podpowiadamy, jak zaprogramować robota po jego zbudowaniu. Dlaczego najpierw zbudować robota? Z dwóch powodów: Po pierwsze, robot oparty na zestawie jest niedrogi w porównaniu z zakupem gotowego robota. Po drugie, budując robota, dowiesz się, jak działa robot i jak możesz go kontrolować za pomocą Pythona. Wybraliśmy robota wzorowanego na naszym przyjacielu Raspberry Pi. Możesz dostać roboty oparte na wielu innych komputerach, w tym na Arduino, ale z tymi mniejszymi komputerami nie możesz wykonać takiego rodzaju przetwarzania lub sztucznej inteligencji, jak na Raspberry Pi.

Przedstawiamy marsjańskiego łazika PiCar-B

Kiedy decydowaliśmy, który robot zbudować w tej książce, przyjrzeliśmy się i zmontowaliśmy cztery różne małe samochody-roboty. Wszystkie były w podobnej cenie i każdy miał jakieś wady. Jednak po dokładnym rozważeniu zdecydowaliśmy się na użycie Adeept PiCar-B.



Zrobiliśmy to z kilku powodów:

-Instrukcja montażu była przejrzysta, zawierała wiele zdjęć i schematów.

-Dostarczone oprogramowanie było kompatybilne z Pythonem 3 (i wersją -Stretch systemu operacyjnego Raspberry Pi).

-PiCar-B nie wymagał lutowania.

- Miał rozsądną cenę i dobrą dostępność.

Samochód sterowany radiem również można uznać za robota. Dlaczego wybraliśmy ten samochód? Dzieje się tak dlatego, że możesz łatwo dostać się do oprogramowania i dodać więcej oprogramowania, aby robot robił to, co chcesz. Tak, ma interfejs użytkownika (patrz następny rozdział) na dodatkowym komputerze. Jednak jesteśmy o wiele bardziej zainteresowani umieszczeniem naszego własnego oprogramowania Pythona w robocie niż zabawą joystickiem.

Co jest potrzebne do budowy

Do zbudowania robota opisanego w tym rozdziale potrzebne są trzy rzeczy, oprócz kilku podstawowych narzędzi (chociaż w zestawie znajdują się klucze imbusowe i śrubokręty) oraz plastikowych opasek zaciskowych do połączenia przewodów po złożeniu:

* Raspberry Pi 3B+: Tak, można sobie poradzić z mniejszym Raspberry Pi (takim jak Raspberry Pi Zero), ale zalecamy zakup szybkiego, aby można było wykonywać bardziej wyrafinowane przetwarzanie na pokładzie robota. Cena, którą płacisz za szybsze Pi (takie jak 3B), dotyczy zużycia energii i żywotności baterii. Dla naszych celów jest to dobry kompromis. Między innymi Raspberry Pi 3B można kupić w

-Amazon.om (upewnij się, że kupiłeś 3B)

-Newark.com

-Adafruit.com

-Sklep.switchdoc.com

* Adeept Raspberry Pi PiCar-B: Adeept Raspberry Pi PiCar-B nie jest tak dostępny jak Raspberry Pi. Kupując to, upewnij się, że kupujesz PiCar-B, a nie PiCar-A. Dodają Mars Rover do nazwy tego produktu w swoim katalogu, więc szukaj „Adeept Mars Rover PiCar-B". Możesz kupić PiCar-B w tych miejscach:

-Amazon.com (https://amzn.to/2B7mtop)

- eBay.com

- Adept.com

- Sklep.switchdoc.com

* Akumulatory LiPo 18650: PiCar-B wymaga dwóch akumulatorów LiPo 18650 3,7 V 5000 mAh. Możesz także zasilić robota, wyłączając wyłącznik zasilania (lub wyjmując baterie) i zasilając Raspberry Pi z wtyczki micro USB, która następnie zasila zarówno robota, jak i Raspberry Pi. Zasilanie robota i Raspberry Pi są ze sobą połączone. Wybrany przez nas pakiet miał dwa komplety baterii i dołączoną ładowarkę ścienną. Możesz kupić tego rodzaju baterie w każdym miejscu, w tym

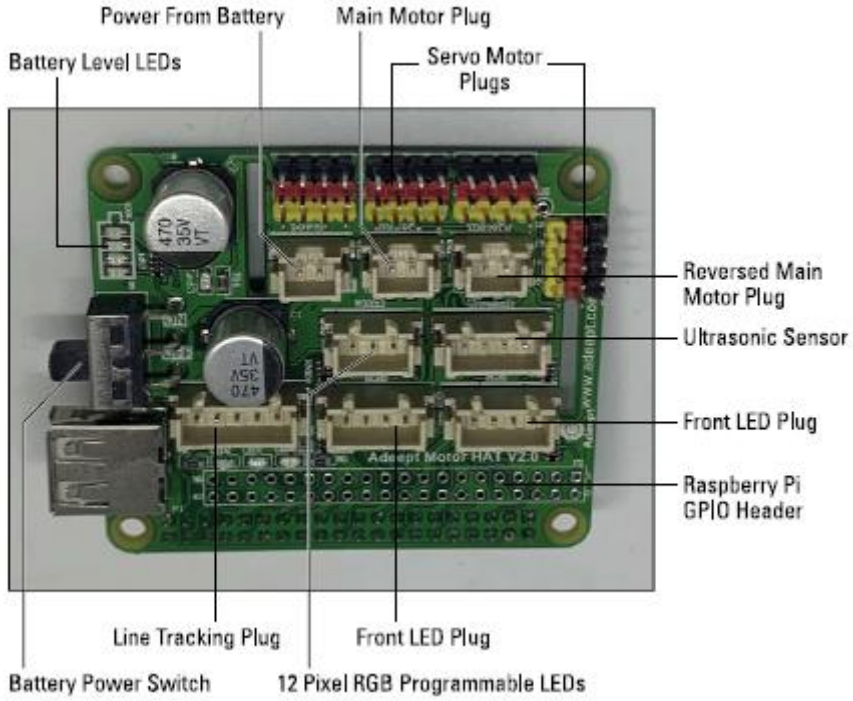
- Amazon.com (https://amzn.to/2TgPsx1)

- Wiele, wiele innych miejsc.

Zrozumienie komponentów robota

Teraz nadszedł czas, aby przyjrzeć się komponentom PiCar-B. Nie będziemy skupiać się na mechanicznej strukturze robotów, ale raczej na każdym z aktywnych komponentów. Porozmawiamy również o oprogramowaniu Python używanym do komunikacji z każdym urządzeniem używanym w oprogramowaniu do testowania systemu Python w dalszej części tego rozdziału oraz w naszym własnym oprogramowaniu robotów w rozdziale 3. Podamy małe fragmenty kodu Pythona, aby pokazać, jak każde czujników i silników są sterowane. Aby uzyskać więcej informacji na temat kodu i opisu, zobacz sekcję „Uruchamianie — testy na łaziku w Pythonie" w dalszej części tego rozdziału.

Płyta kontrolera



Główne dwa układy na płycie to -PCA9685, urządzenie I2C używane do sterowania maksymalnie 16 serwomotorami (z których trzy są używane, więc jest dużo miejsca na rozbudowę) oraz L289P , który służy do zasilania głównego silnika napędowego. Pozostała część płytki służy do podłączenia pinów GPIO (wejście-wyjście ogólnego przeznaczenia) z Raspberry Pi do różnych czujników i urządzeń. Posiada również zasilacz 5,0 V, który zasila Raspberry Pi i silniki z akumulatorów LiPo.

Silniki serwo

Serwomotory są ogólnie połączeniem trzech rzeczy: silnika prądu stałego, prostego obwodu sterującego i zestawu przekładni. Czasami znajdziesz potencjometr (który jest zmiennym rezystorem), który daje informację zwrotną o położeniu. Położeniem sterujesz za pomocą modulacji szerokości impulsu (PWM), techniki, którą widzieliśmy w rozdziale 1 do kontrolowania jasności diody LED. - Mikrosilnik serwo SG90 dostarczany z robotem to mały, niedrogi serwosilnik dostępny z wielu źródeł. Ma napięcie robocze 3,0–7,0 V przy poborze prądu około 40 mA (40 miliamperów; miliamper to 1/1000

ampera prądu) maksymalnie, więc 5 V na Raspberry Pi jest dobre do obsługi tego serwomechanizmu. Może obrócić się o około 90 stopni w każdym kierunku, co daje w sumie 180 stopni. W większości serwomotorów są trzy przewody sterujące, a SG90 jest bez wyjątku. Te trzy przewody to:

- Żółty: sygnał sterujący PWM

- Czerwony: Zasilanie (5 V, w naszym przypadku)

- Brązowy: ziemia

Serwa są zasilane w sposób ciągły i generalnie mają zakres ruchu tylko około 180–270 stopni. Wszystkie trzy serwosilniki to mikro serwa SC-90 9g. Te serwa są sterowane z układu PCA9685 I2C, więc nie wykorzystują żadnych linii GPIO z Raspberry Pi. Znajduje się na magistrali I2C Raspberry Pi.

Aby sterować serwomotorem, musimy po prostu ustawić wartość na pozycję, w której chcemy, aby serwo przesunęło się na odpowiedniej linii PWM w PCA9685:

```
print ("--------------------")

print ("Servo Test - Head Left")

print ("--------------------")

pwm.set_pwm(HEAD_TURN_SERVO, 0, calValues.look_left_max)

time.sleep(1.0)
```

Liczba, którą przekazujemy do serwosilnika w celu określenia pozycji (calValues.look_left_max) jest określana empirycznie i jest ustawiana na podstawie zasięgu serwomotoru, gdy wydajesz mu polecenia w lewo i w prawo. Zobacz „Kalibrowanie serwomechanizmów” w dalszej części tego rozdziału.

Silnik napędowy

Głównym silnikiem napędowym jest silnik prądu stałego z dwoma przewodami: zasilającym i uziemiającym.

Po podłączeniu zasilania (na przykład poprzez podłączenie 5 V do linii zasilającej) silnik zacznie się obracać. Odwróć przewody zasilające i uziemiające, a silnik zacznie się obracać w przeciwnym kierunku. Kontrolujesz prędkość silnika prądu stałego za pomocą modulacji szerokości impulsu (PWM), techniki kontrolowania jasności diody LED (patrz rozdział 1 tego minibooka). Jeśli moc zostanie zmieniona na 50 procent (połowa włączenia/połowa wyłączenia), silnik będzie obracał się z połową prędkości. Czasami umieszczasz „enkoder” na wale silnika, który pozwala ci odczytać z komputera, jak daleko wał się obrócił, dając komputerowi informacje zwrotne, które mogą być przydatne. Ten silnik wykorzystuje

sześć linii GPIO z Raspberry Pi do sterowania prędkością i kierunkiem. Zawiłości sterowania tym silnikiem są dobrze ukryte przed użytkownikiem. Oto kod Pythona, aby przesunąć samochód do przodu:

motor.motor_left(MOTOR_START, forward,left_spd*spd_ad)

motor.motor_right(MOTOR_START,backward,right_spd*spd_ad)

Dlaczego włączamy lewy i prawy silnik, skoro w PiCar-B jest tylko jeden silnik napędowy? Powodem jest to, że nie możesz być pewien, w jaki sposób podłączony jest twój silnik; może być podłączony w jednym kierunku lub może być odwrócony. (Nasza była odwrócona.) Masz dwie wtyczki silnika na płycie kontrolera. Jeśli polecenie do przodu spowoduje ruch robota do tyłu, wystarczy przesunąć silnik do drugiego złącza silnika i wszystko działa. Napisanie poprzedniego kodu (przy użyciu obu sterowników silnika) umożliwia oprogramowaniu pracę z dowolnym rodzajem silnika.

Dioda RGB

Przód robota ma dwie pojedyncze diody LED RGB, po jednej z każdej strony.

Te duże diody LED mają w rzeczywistości trzy diody LED wewnątrz obudowy. Diody LED są czerwone, niebieskie i zielone i są indywidualnie sterowane przez linie GPIO (uruchamiający programowy kod PWM, który pozwala nam mieszać diody R, G i B). Każda z diod RGB wykorzystuje trzy linie GPIO z Raspberry Pi:

print ()

print ("--------------------")

print ("Left Front LED Test - Red ")

print ("--------------------")

led.side_on(led.left_R)

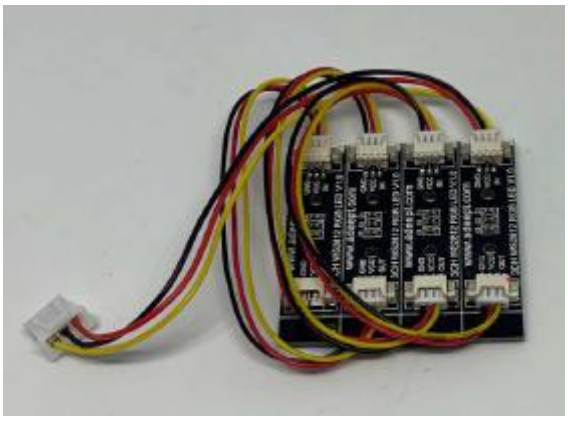
time.sleep(1.0)

led.side_off(led.left_R)

Włączasz diody LED indywidualnie i możesz (przy użyciu innego oprogramowania) faktycznie ustawić jasność każdej diody LED.

Programowalne diody LED Pixel RGB

Na robocie znajduje się 12 programowalnych diod LED RGB, dwa zestawy po trzy na spodzie robota i dwa zestawy po trzy skierowane do tyłu. (Patrz rysunek 2-6.)



Tych 12 diod LED jest połączonych szeregowo, jak lampki choinkowe. I tak jak niektóre lampki choinkowe, jeśli jedna zgaśnie, cała reszta sznurka też zgaśnie. Dzieje się tak, ponieważ są one kontrolowane przez pojedynczy strumień danych szeregowych, który jest przesyłany przez wszystkie światła przez Raspberry Pi. Ten strumień szeregowy jest bardzo precyzyjnie synchronizowany, co wymaga specjalnego oprogramowania na Raspberry Pi, aby działał. Ciąg Pixel wykorzystuje pojedynczy pin GPIO pochodzący z Raspberry Pi:

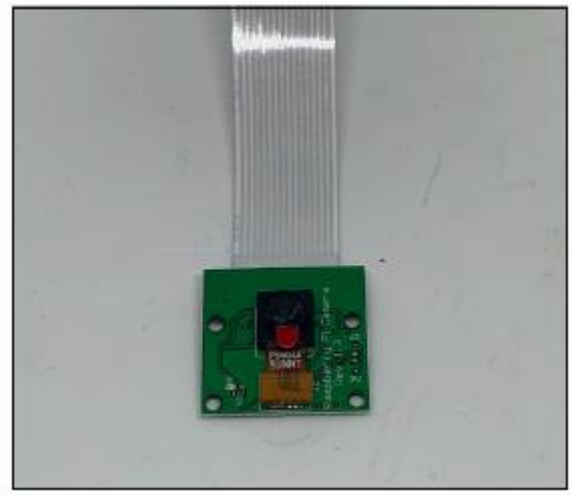
```
print ()

print ("--------------------")

print ("12 RGB Pixel LED Test - On ")

print ("--------------------")

rainbowCycle(strip, wait_ms=20, iterations=3)
```

To polecenie przełącza tęczę kolorów wokół wszystkich 12 diod LED z tyłu i na spodzie robota. Sterownik dla tych pikseli RGB jest bardzo skomplikowany, ale dostarczymy kod do łatwego sterowania diodami LED i podamy kilka przykładów wykorzystania ich do innych celów. Naprawdę kochamy te diody LED i używamy ich w wielu projektach.

STRUMIEŃ PIXELI RGB NA RASPBERRY PI

Raspberry Pi ma złożony, wielopłaszczyznowy system operacyjny oparty na systemie Linux. Jest to wielozadaniowy system operacyjny z wywłaszczaniem, co oznacza, że praktycznie każde zadanie (i wszystkie zadania użytkownika) może zostać przerwane (czyli zatrzymane), a tym samym nasz strumień szeregowy do diod LED Pixela zatrzymany i do pewnego stopnia uszkodzony. Biblioteka, której używamy, rozwiązuje problem sterowania w czasie rzeczywistym, wykorzystując sprzętowe PWM i DMA na procesorze Raspberry Pi. Moduł PWM (modulacja szerokości impulsu) może generować sygnał o określonym współczynniku wypełnienia; na przykład, aby sterować serwomechanizmem lub przyciemnić diodę LED. Moduł DMA (bezpośredni dostęp do pamięci) może przesyłać bajty pamięci między częściami procesora bez użycia procesora. Używając DMA do wysyłania określonej sekwencji bajtów do modułu PWM, sygnał danych Pixel może być generowany bez przerywania przez system operacyjny Raspberry Pi. Ponieważ procesory typu Arduino tak naprawdę nie mają systemu operacyjnego, generowanie tych sygnałów na Arduino jest dość łatwe w porównaniu z Raspberry Pi.

Procesory takie jak ESP8266 i ESP32 mają zadania działające w tle (takie jak Wi-Fi), dlatego wymagają specjalnych sterowników, aby to zrekompensować, aby uniknąć uszkodzenia danych i migotania. Należy zauważyć, że chociaż działa to dobrze na Raspberry Pi 3B, diody LED nie zawsze działają dobrze ze starszymi i mniejszymi Raspberry Pis (na przykład A, 3B, Pi Zero lub Pi Zero W). Kamera zastosowana w PiCar-B to klasyczna kamera Raspberry Pi w wersji 2.1.



Ma 8-megapikselowy czujnik firmy Sony i może obsługiwać zdjęcia w rozdzielczości do 3280 x 2464 przy 15 klatkach na sekundę. Ma dobrą reakcję na kolory i jest dobrze obsługiwany w Pythonie i OpenCV (OpenCV to pakiet Open Source Computer Vision, którego używamy w dalszej części rozdziału 4). Kamera Pi komunikuje się z Raspberry Pi za pośrednictwem równoległego kabla taśmowego bezpośrednio do płyty Raspberry Pi.

Poniższy kod otwiera małe okno w interfejsie GUI Raspberry Pi, czeka 20 sekund, przesuwa okno i zmienia jego rozmiar, czeka 2 sekundy, przesuwa je ponownie, a następnie zamyka okno:

```
print ()

print ("--------------------")

print ("Open Video Window")

print ("--------------------")

camera.resolution = (1024, 768)

camera.start_preview(fullscreen=False,window=

(100,100,256,192))

time.sleep(20)

camera.preview.window=(200,200,256,192)

time.sleep(2)a

camera.preview.window=(0,0,512,384)

time.sleep(2)
```

camera.close()

Jeśli używasz VNC, aby zobaczyć GUI Raspberry Pi (tak jak my), musisz otworzyć opcje serwera VNC w prawym górnym rogu, przejść do Opcji, a następnie do Rozwiązywanie problemów i kliknąć Włącz Pole wyboru Tryb bezpośredniego przechwytywania. Patrz sekcja „Montaż robota” w dalszej części tego rozdziału.

Czujnik ultradźwiękowy

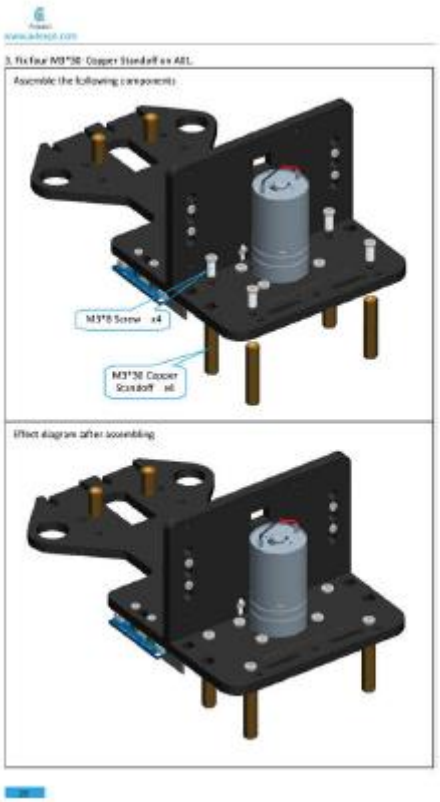
Detektor ultradźwiękowy zastosowany w PiCar-B to bezkontaktowy moduł pomiaru odległości, który pracuje z częstotliwością 40KHz.

Po dostarczeniu impulsowego sygnału wyzwalającego o czasie dłuższym niż 10 uS przez pin sygnałowy, urządzenie emituje 8 cykli o poziomie cyklu 40 kHz i wykrywa echo. Szerokość impulsu sygnału echa jest proporcjonalna do zmierzonej odległości. Proste, ale skuteczne. Stosowany wzór to: Odległość = najwyższy czas sygnału echa * Prędkość dźwięku (340M/S)/2. W poniższym kodzie wywołanie ultra.checkdisk() wywołuje oprogramowanie, które ustawia bit transmisji GPIO, a następnie czeka na powrót echa, oznaczając czas odebrania. Następnie oblicza czas przejazdu i podaje odległość w metrach, którą przeliczamy na centymetry.

```
print ()

print ("--------------------")

print ("Ultrasonic Distance Test")

print ("--------------------")

average_dist = 0.0

for i in range(0,10):

distance = ultra.checkdist()

average_dist = average_distance distance

print ("Distance = {:6.3f}cm ".format( distance*100))

time.sleep(1.0)

average_distance = average_distance / 10

print ("average_dist={:6.3f}cm".format(average_dist*100))
```

Składanie robota

PiCar-B jest dostarczany z instrukcją montażu wraz z wydmuchanymi zdjęciami pokazującymi, jak wszystko idzie razem.



Złożenie robota i przejście do punktu rozpoczęcia testów zajęło nam około czterech godzin.

Uwaga: Nie wychodź poza rozdział 2 w instrukcji montażu PiCar-B. Rozdział 3 rozpoczyna instalację całego oprogramowania Adeept na Raspberry Pi i będziesz chciał przetestować robota za pomocą oprogramowania z tej książki, aby lepiej zrozumieć części robota przed kontynuowaniem.

Więc idź zbudować swojego robota, a potem spotkaj się z nami tutaj, aby rozpocząć testowanie nowego robota. Kalibracja serwomechanizmów to pierwszy krok! Następnie przeprowadzamy pełny test systemu. Rysunek przedstawia zmontowanego robota.

Oto kilka pomocnych wskazówek dotyczących budowania robota:

Po lewej stronie płytki napędu silnika (patrząc od przodu robota) znajduje się włącznik zasilania. W instrukcji nie jest to wspomniane. Odcina zasilanie z akumulatorów.

Upewnij się, że używasz dłuższego z dostarczonych kabli taśmowych do kamery Pi. Krótki nie dosięgnie. Zmień go przed zainstalowaniem kamery w robocie.

Poprowadź przewody, jak pokazano na rysunku 2-10, aby silniki się nie zakleszczyły. Użyj plastikowych opasek drucianych, aby utrzymać rzeczy na miejscu, pozostawiając miejsce na obracanie serwomechanizmów i obudowy. W zestawie znajduje się opaska druciana, która zakryje niektóre przewody.

Zwróć szczególną uwagę na orientację części plastikowych i serwomechanizmów podczas montażu. Prawie wszystkie z nich mają asymetryczną górę i dół i muszą być odpowiednio ustawione, aby można je było zmontować.

Nie upuszczaj małych śrubek na dywan. Na pewno trudno je znaleźć!

PRAWIDŁOWE WYŁĄCZANIE RASPBERRY PI

W przeciwieństwie do większości innych komputerów, Raspberry Pi nie ma włącznika/wyłącznika. Jednak, podobnie jak w przypadku wielu innych komputerów, samo wyciągnięcie wtyczki z Raspberry Pi może mieć tragiczne konsekwencje, w tym przypadku uszkodzenie karty SDCard, której Raspberry Pi używa do przechowywania programów i danych. Przed wyłączeniem Raspberry Pi wpisz w oknie terminala: sudo halt. To bezpiecznie wyłącza Raspberry Pi. Po uruchomieniu tego polecenia po chwili zobaczysz, że lampka „ACT” (zielona) miga 10 razy (w odstępach 0,5 sekundy). Kiedy przestanie migać, zielone światło zgaśnie. W tym momencie można bezpiecznie odłączyć zasilanie lub wyciągnąć wtyczkę. Czerwona dioda LED zasilania pozostanie włączona, dopóki Raspberry Pi będzie zasilane.

Kalibracja serwomechanizmów

Teraz, gdy zmontowałeś robota, nadszedł czas, aby rozpocząć testowanie rzeczy. Pierwszą rzeczą do zrobienia jest skalibrowanie serwomotorów. Dlaczego potrzebują kalibracji? Ponieważ chociaż instrukcje nakazują pozostawienie serwomechanizmów wyśrodkowanych podczas montażu, niekoniecznie będą one całkowicie we właściwym miejscu. Program calibrateServo.py uruchamia każde z trzech serwomechanizmów od jednego końca do drugiego. Obserwując obracające się silniki, możesz zapisać maks., min. i środek każdego serwomechanizmu z wyświetlacza w oknie terminala. Potem umieszczasz te wartości w programie calValues.py, aby pozostałe programy miały do nich dostęp. Wartości w calValues.py są odpowiednie dla naszego robota i prawdopodobnie będą zbliżone do twoich, ale dla pewności powinieneś uruchomić program. Kod calibrateServo jest następujący:

```
#!/usr/bin/python3

# calibrate servos

import time

import Adafruit_PCA9685

import calValues

#import the settings for servos

pwm = Adafruit_PCA9685.PCA9685()

pwm.set_pwm_freq(60)

#servo mapping

# pmw 0 head tilt
```

```
HEAD_TILT_SERVO = 0
# pwm 1 head turn
HEAD_TURN_SERVO = 1
# pwm 2 wheels turn
WHEELS_TURN_SERVO = 2
if __name__ == '__main__':
print("--------------------")
print("calibrate wheel turn")
print("--------------------")
for i in range(calValues.turn_right_max,
calValues.turn_left_max,10):
pwm.set_pwm(WHEELS_TURN_SERVO,0, i)
print("servoValue = ", i)
time.sleep(0.5)
print("--------------------")
print("calibrate head turn")
print("--------------------")
for i in range(calValues.look_right_max,
calValues.look_left_max,10):
pwm.set_pwm(HEAD_TURN_SERVO,0, i)
print("servoValue = ", i)
time.sleep(0.5)
print("--------------------")
print("calibrate head up/down")
print("--------------------")
for i in range(calValues.look_up_max,
calValues.look_down_max,10):
pwm.set_pwm(HEAD_TILT_SERVO,0, i)
print("servoValue = ", i)
time.sleep(0.5)
```

The code is pretty straight forward, but let me talk about one of the servo

program loops.

```
for i in range(calValues.look_right_max,

calValues.look_left_max,10):

pwm.set_pwm(HEAD_TURN_SERVO,0, i)

print("servoValue = ", i)

time.sleep(0.5)
```

Ta pętla przechodzi przez zakres serwomechanizmów podany w calValues.py od prawej do lewej, obracając głowicę w krokach co 10. Daje to całkiem niezły pomysł, gdzie powinny znajdować się prawa, lewa i środek (biorąc pod uwagę rama robota też!) i możesz dodać te wartości do calValues.py. Plik calValues.py zawiera wartości kalibracyjne dla serwomotorów. Zastępujesz wartości w tym programie własnymi wartościami z calibrate Servos.py:

```
# Servo calibration values

# head

look_up_max = 150

look_down_max = 420

look_tilt_middle = 330

# head turn

look_right_max = 200

look_left_max = 450

look_turn_middle = 310

# wheels

turn_right_max = 180

turn_left_max = 460

turn_middle = 320

# turn_speed

look_turn_speed = 5

# motor speed

left_spd = 100 #Speed of the car

right_spd = 100 #Speed of the car
```

Uruchamianie testów na łaziku w Pythonie

Dobra, teraz masz zmontowany PiCar-B, czas na kilka testów. Jeśli zainstalowałeś oprogramowanie Adeept na swoim Raspberry Pi, musisz wyłączyć automatyczne uruchamianie ich oprogramowania. W tym celu zmień następujący wiersz w pliku ~/.config/autostart/car.desktop:

Exec=sudo python3 /home/Adeept_PiCar-B/server/server.py

Do

#Exec=sudo python3 /home/Adeept_PiCar-B/server/server.py

A następnie zrestartuj swoje pi za pomocą: Sudo reboot. Oczywiście mówiliśmy ci, abyś nie instalował oprogramowania, ale jeśli byłeś zbyt podekscytowany i zrobiłeś to, musisz zrobić powyższe, w przeciwnym razie będziesz mieć konflikty w oprogramowaniu. Oto film przedstawiający działanie następującego oprogramowania testowego Pythona na robocie PiCar-B: https://youtu.be/UvxRBJ-tFw8.

To właśnie będziesz wkrótce biegać.

Instalowanie oprogramowania do testu CarPi-B Python

Przede wszystkim przejdź do strony wsparcia tej książki na www.dummies.com (patrz Wprowadzenie) i pobierz oprogramowanie dla Księgi 7, Rozdział 2. Przejdź do katalogu Testing, a następnie wykonaj te instrukcje, które są niezbędne do uzyskania 12 programowalnych diod RGB do pracy na Raspberry Pi:

1. Zainstaluj kilka bibliotek programistycznych, które pozwolą nam skompilować oprogramowanie. Jest to instalowane przy użyciu normalnego instalatora Raspberry Pi w następujący sposób:

sudo apt-get install build-essential python3-dev git scons swig

2. Pobierz kod neopixel z github za pomocą polecenia clone, które skopiuje cały kod źródłowy na komputer lokalny:

git clone https://github.com/jgarff/rpi_ws281x.git

3. Przejdź do tego katalogu i uruchom scons, aby skompilować oprogramowanie:

cd rpi_ws281x

scons

4. Przejdź do katalogu python i stamtąd zainstaluj moduł Pythona:

cd python

5. Zainstaluj plik biblioteki Pythona 3, używając:

sudo python3 setup.py install

Kod testowy PiCar-B Pythona

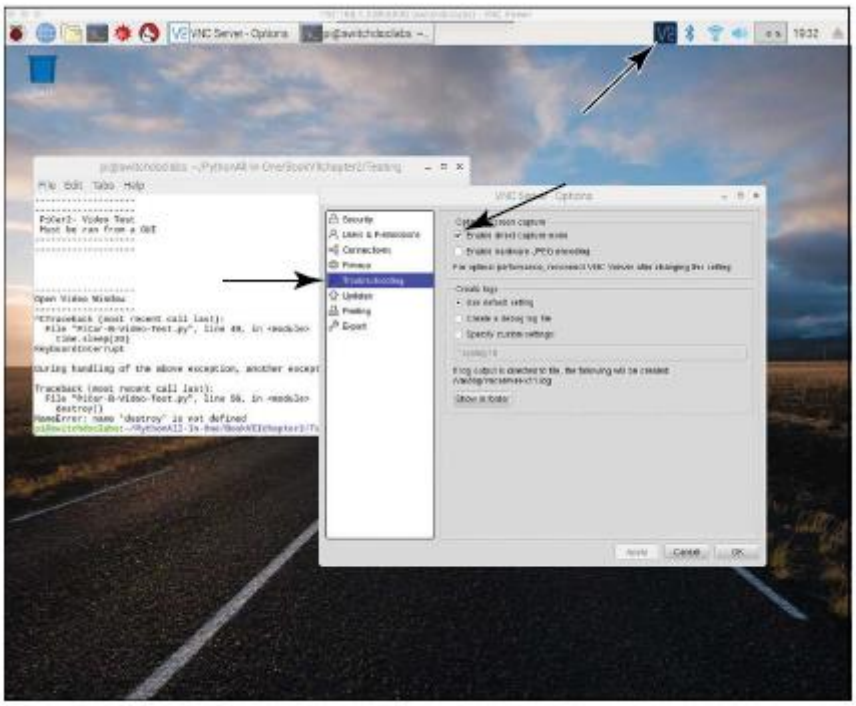
Plik ma około 370 wierszy, więc nie podajemy tutaj pełnej listy. Ważne części tego pliku zostały omówione powyżej wraz z poszczególnymi komponentami i czujnikami. W katalogu Testing znajduje się wiele innych bibliotek i plików. Jeśli jeszcze tego nie zrobiłeś, przejdź do strony wsparcia tej książki pod adresem www.dummies.com (patrz Wprowadzenie) i pobierz oprogramowanie do książki 7, rozdział 2. Uruchom oprogramowanie testowe, wpisując:

sudo python3 PiCar-B-Test.py

Powinieneś natychmiast zobaczyć, jak Twój samochód zaczyna wykonywać sekwencję testową, jak pokazano na https://youtu.be/UvxRBJ-tFw8.

Testowanie wideo z kamery Pi

Aby przygotować się do tego testu, musisz uruchomić GUI na swoim Raspberry Pi. Jeśli używasz VNC do wyświetlania GUI na innym komputerze, musisz włączyć opcję VNC. Otwórz opcje serwera VNC w prawym górnym rogu, przejdź do Opcji, a następnie do Rozwiązywanie problemów i kliknij pole wyboru Włącz tryb bezpośredniego przechwytywania.



To oprogramowanie testowe (PiCar-B-Video-Test.py) jest następujące:

```
#!/usr/bin/python3

DEBUG = True

VIDEOTEST = True

# runs through a video tests for the PiCar-B

import RPi.GPIO as GPIO

import motor

import ultra

import socket

import time

import threading

import turn

import led

import os

import picamera

from picamera.array import PiRGBArray
```

```
import cv2
import calValues
if __name__ == '__main__':
camera = picamera.PiCamera() #Camera initialization
camera.resolution = (640, 480)
camera.framerate = 7
rawCapture = PiRGBArray(camera, size=(640, 480))
try:
print ("--------------------")
print ("--------------------")
print (" PiCar2- Video Test")
print (" Must be run from a GUI")
print ("--------------------")
print ("--------------------")
print ()
print ()
if (VIDEOTEST):
print ()
print ("--------------------")
print ("Open Video Window")
print ("--------------------")
camera.resolution = (1024, 768)
camera.start_preview(fullscreen=False,
window=(100,100,256,192))
time.sleep(20)
camera.preview.window=(200,200,256,192)
time.sleep(2)
camera.preview.window=(0,0,512,384)
time.sleep(2)
camera.close()
except KeyboardInterrupt:
```

destroy()

Ten kod otwiera małe okno w GUI Raspberry Pi, czeka 20 sekund, przesuwa okno i zmienia jego rozmiar, czeka 2 sekundy, przesuwa je ponownie, a następnie zamyka okno. Teraz skończyłeś budować i testować swojego robota. Następnie dodamy mózgi Pythona i trochę pobawimy się robotem.