Wyczuwanie świata za pomocą Pythona: świat I2C

Zanim przejdziemy do tego, jak wyczuwać świat w Pythonie, przejrzyjmy kilka problemów sprzętowych. Możesz pominąć to wszystko i nadal używać Pythona do komunikacji z tymi urządzeniami, oczywiście, ale dobrze jest mieć pewne doświadczenie. Zawsze możesz wrócić do tego później, gdy będziesz mieć trochę doświadczenia z tymi urządzeniami. Dostępne czujniki dla Raspberry Pi i innych małych komputerów są w tysiącach. Od wykrywania ludzi przed komputerem (PIR) po wykrywanie niezliczonych warunków środowiskowych (temperatura/wilgotność/jakość powietrza/itp.), istnieje wiele niedrogich sposobów na monitorowanie świata fizycznego przez komputer. Jak zawsze, najważniejszą rzeczą, którą musisz wiedzieć o tych czujnikach, jest to, jak możesz rozmawiać z nimi za pomocą komputera, co zwykle odbywa się za pośrednictwem interfejsu. Interfejs składa się z dwóch elementów: interfejsu sprzętowego, który zawiera piny, typy i poziomy napięcia, oraz interfejsu programowego, który jest zwykle nazywany sterownikiem lub interfejsem API (interfejs programowania aplikacji). Istnieją cztery główne sposoby uzyskiwania danych z czujników zewnętrznych do komputera:

Wejście cyfrowe - piny GPIO zaprogramowane jako linie wejściowe.

Cyfrowe wejście analogowe — wartości analogowe, które muszą przejść przez przetwornik analogowo-cyfrowy (ADC), aby mogły zostać odczytane przez komputer.

Cyfrowy I2C (czyt. I-kwadrat-C) (układ międzyscalony) bu

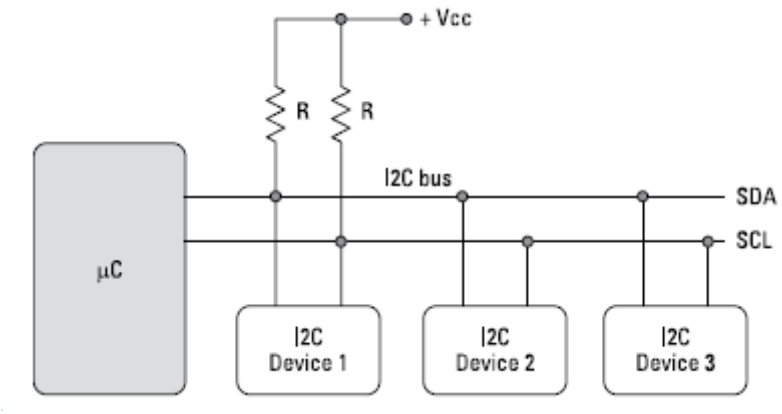
Cyfrowy SPI (szeregowy interfejs urządzeń peryferyjnych)

Zajmujemy się czujnikami wykorzystującymi wejścia cyfrowe, wejścia analogowe oraz interfejsy I2C. Dlaczego nie SPI? Tylko dla uproszczenia. Większość części SPI ma również interfejs I2C na chipie, a większość małych płyt komputerowych ma wbudowany interfejs I2C.

Zrozumienie I2C

Pierwszą rzeczą, którą należy wiedzieć o I2C, jest to, że każde urządzenie na magistrali I2C ma adres. Na przykład adres czujnika temperatury i wilgotności HDC1080, którego używamy w tym rozdziale, ma adres 0x40. Co oznacza „0x” w tym adresie? Oznacza to, że następująca po nim liczba jest zapisana w systemie szesnastkowym, podstawa 16 zamiast podstawy 10 (nasz normalny system liczbowy). Aby zrozumieć ten interfejs, przyjrzyjmy się, czym jest magistrala I2C. Magistrala I2C jest często używana do komunikacji z układami scalonymi lub czujnikami, które znajdują się na tej samej płycie lub znajdują się fizycznie blisko procesora. I2C został po raz pierwszy opracowany przez firmę Phillips (obecnie NXP Semiconductors). Aby często obejść problemy z licencjami (które w dużej mierze zniknęły) magistrala będzie się nazywać TWI (Two Wire Interface). SMBus, opracowany przez firmę Intel, jest podzbiorem I2C, który ściślej określa protokoły. Nowoczesne systemy I2C przejmują zasady i reguły z SMBus, czasami obsługując oba przy minimalnej wymaganej rekonfiguracji. Zarówno Arduino, jak i Raspberry Pi obsługują magistralę I2C. I2C zapewnia dobre wsparcie dla powolnych, bliskich urządzeń peryferyjnych, które wymagają adresowania tylko okazjonalnie. Na przykład urządzenie do pomiaru temperatury na ogół zmienia się bardzo powoli, dlatego jest dobrym kandydatem do użycia I2C, podczas gdy kamera szybko generuje dużo danych i potencjalnie często się zmienia. I2C wykorzystuje tylko dwie dwukierunkowe linie z otwartym drenem (otwarty dren oznacza, że urządzenie może obniżyć poziom do masy, ale nie może podciągnąć linii do Vdd. Stąd nazwa otwarty dren. Zatem wymaganiem magistrali I2C jest, aby obie linie są podciągnięte do Vdd. Jest to ważny obszar i niewłaściwe podciągnięcie linii jest pierwszym i najczęstszym błędem popełnianym przy pierwszym użyciu magistrali I2C. Płytka Pi2Grover, której używamy w tej książce, zawiera rezystory podciągające

10 kΩ, więc nie powinieneś się tym martwić. Dwie linie to SDA (linia danych szeregowych) i SCL (linia zegara szeregowego). Istnieją dwa rodzaje urządzeń, które można podłączyć do magistrali I2C: urządzenia nadrzędne i urządzenia podrzędne. mieć jedno urządzenie Master (w naszym przypadku Raspberry Pi) i wiele urządzeń Slave, każde z własnym 7-bitowym adresem .



Gdy jest używany z Raspberry Pi, Raspberry Pi działa jako Master, a wszystkie inne urządzenia są podłączone jako Slave. Protokół I2C wykorzystuje trzy rodzaje komunikatów:

Cyfrowa pojedyncza wiadomość, w której master zapisuje dane do slave'a

Cyfrowa pojedyncza wiadomość, w której urządzenie nadrzędne odczytuje dane z urządzenia podrzędnego

Cyfrowe komunikaty łączone, w których urządzenie nadrzędne wysyła co najmniej dwa odczyty i/lub zapisy do jednego lub większej liczby urządzeń podrzędnych

Na szczęście dla nas większość złożoności obsługi magistrali I2C jest ukryta przez sterowniki i biblioteki Pythona.

Eksploracja I2C na Raspberry Pi

Pierwszą rzeczą, którą chcesz zrobić na swoim Raspberry Pi, jest poznanie okna terminala, wiersza poleceń i edytorów tekstu. Jeśli jeszcze tego nie zrobiłeś, zapoznaj się z częścią 1 tego minibooka. Aby korzystać z magistrali I2C na Raspberry Pi, musisz upewnić się, że jest ona włączona w systemie operacyjnym. Oto dobry samouczek od Adafrui9t, jak to zrobić: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpiosetup/configuring-i2c. Czy zrobiłeś to dobrze? Łatwym sposobem sprawdzenia tego jest wpisanie następującego polecenia w oknie terminala:

I2cdetect -y 1

Jeśli powróci:

-bash: i2cdetect: command not found

W takim razie nie włączyłeś swojej magistrali I2C. Powtórz samouczek, aby to naprawić. Z drugiej strony, jeśli powróci:

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          --                                  
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

W takim razie odniosłeś sukces! Zauważ, że wszystkie kreski oznaczają, że na szynie I2C nie ma czujników. W następnej sekcji dodamy prosty. Porozmawiajmy teraz o tym, jak komunikować się z urządzeniami I2C w Pythonie.

Rozmowa z urządzeniami I2C za pomocą Pythona

Aby rozmawiać z urządzeniem I2C, powinieneś mieć je na magistrali. Dobrym na początek jest czujnik temperatury i wilgotności HDC1080.

Możesz kupić jeden z tych niedrogich czujników na store.switchdoc.com lub na amazon.com.

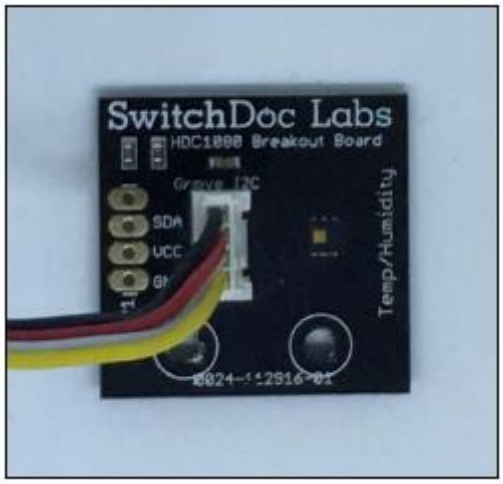
CZUJNIK TEMPERATURY I WILGOTNOŚCI TEXAS INSTRUMENTS HDC1080

To całkiem niesamowite urządzenie, biorąc pod uwagę, jak niedrogie jest. HDC1080 to czujnik temperatury i wilgotności kompatybilny z HDC1000. Znajduje się pod adresem I2C 0x40. Czujnik temperatury i wilgotności Grove (HDC1080) wykorzystuje czujnik HDC1080 firmy Texas Instruments. Jest to cyfrowy czujnik wilgotności ze zintegrowanym czujnikiem temperatury, który zapewnia doskonałą dokładność pomiaru przy bardzo małej mocy. Urządzenie mierzy wilgotność w oparciu o nowatorski czujnik pojemnościowy. Czujniki wilgotności i temperatury są skalibrowane fabrycznie. Innowacyjny pakiet WLCSP (wafer level chip scale package) upraszcza projektowanie płytek przy użyciu ultrakompaktowej obudowy. HDC1080 działa w pełnym zakresie temperatur od –40°C do 125°C i w zakresie 0–100% wilgotności względnej. Dokładność chipa wynosi / 3 procent wilgotności względnej i / 0,2 C dla temperatury.

Uwaga: jeśli kupisz taki na Amazon, będziesz potrzebować kabla krosowego żeńskiego do Grove, jak omówiono w rozdziale 2 tego minibooka. SwitchDoc Labs HDC1080 jest już wyposażony w złącze Grove. Będziesz także potrzebował konwertera Pi2Grover Raspberry Pito-Grove, który został opisany w rozdziałach 1 i 2 tego minbooka, który jest również dostępny na store.switchdoc.com lub na amazon.com. Teraz zainstalujmy czujnik HDC1080 I2C na naszym Raspberry Pi. Wykonaj następujące kroki:

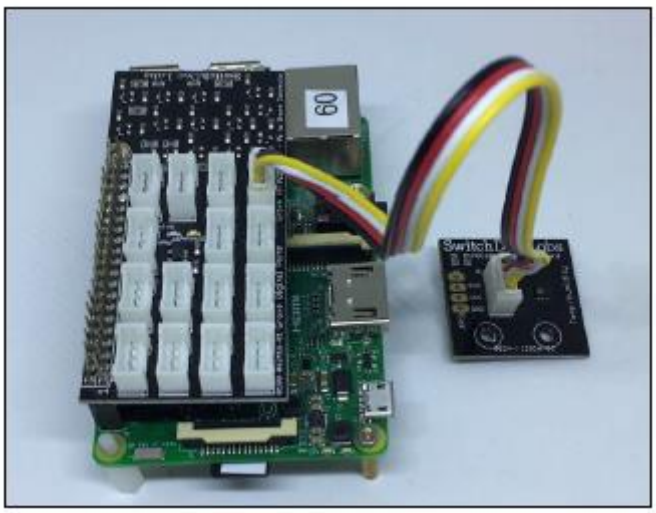
1. Wyłącz Raspberry Pi. Gdy żółta dioda LED przestanie migać, odłącz zasilanie od Raspberry Pi. Nigdy nie podłączaj ani nie wyciągaj niczego z Raspberry Pi bez wyłączania komputera. Wyjątkiem są porty USB, kable audio i kable Ethernet, które są zaprojektowane do obsługi „hot-plug”. Reszta Raspberry Pi nie.

2. Podłącz kabel Grove do HDC1080.



Używamy SwitchDoc Labs HDC1080; jeśli używasz urządzenia Amazon, zapoznaj się z rozdziałem 2 tego minibooka, aby uzyskać informacje na temat używania kabla krosowego Grove. Zawsze wyłączaj Raspberry Pi, najpierw wpisując sudo halt w wierszu poleceń (lub wybierając opcję Shutdown z menu GUI). Poczekaj, aż żółta dioda LED na Raspberry Pi przestanie migać przed odłączeniem przewodu zasilającego. Dzięki temu karta SDCard w Raspberry Pi została przygotowana do wyłączenia i nie uszkodzisz jej. Samo odłączenie Raspberry Pi może nie uszkodzić karty, ale odłączenie go bez wyłączania zwiększa prawdopodobieństwo uszkodzenia. Uszkodzenie karty SDCard może nie być śmiertelne, ale jej naprawa to długi, techniczny i irytujący proces.

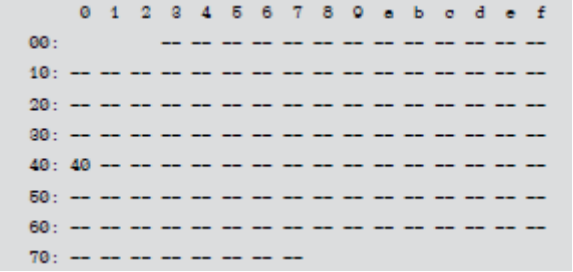
3. Podłącz drugi koniec kabla Grove do jednego ze złączy Grove oznaczonych jako I2C na Pi2Grover, który jest podłączony na górze twojego Raspberry Pi.



Uwaga: I2C to magistrala, co oznacza, że możesz użyć dowolnego z czterech złączy I2C.

4. Włącz Raspberry Pi i otwórz okno terminala.

5. Wpisz w terminalu sudo i2cdetect -y 1, a zostaniesz nagrodzony tym:



Pamiętasz adres 0x40 HDC1080? Tam jest na wyjściu powyżej.

Teraz jesteśmy gotowi, aby przystąpić do używania Pythona do odczytu temperatury i wilgotności z tego czujnika.

Odczyt temperatury i wilgotności z urządzenia I2C za pomocą Pythona

Korzystanie z bibliotek Pythona jest kluczem do produktywnego pisania aplikacji Pythona. Będziemy używać DL_Pi_HDC1080_Python3, dostępnego na github.com. Aby odczytać temperaturę i wilgotność, wykonaj następujące czynności:

1. Najpierw utwórz katalog w swoim katalogu głównym:

cd

mkdir I2CTemperature

cd I2CTemperature

Teraz jesteś w katalogu I2CTemperature.

2. Zanim spojrzysz na kod Pythona do odczytu temperatury, zainstaluj bibliotekę na naszym Raspberry Pi. Robisz to poprzez „klonowanie” biblioteki znajdującej się na github.com za pomocą następującego polecenia w oknie terminala:

git clone https://github.com/switchdoclabs/SDL_Pi_HDC1080_Python3.git

pi@RPi3-60:~/I2CTemperature $ ls

SDL_Pi_HDC1080_Python3

pi@RPi3-60:~/I2CTemperature $

3. Używając nano (lub swojego ulubionego edytora tekstu), otwórz plik o nazwie temperature Test.py i wprowadź następujący kod:

import sys

sys.path.append('./SDL_Pi_HDC1080_Python3')

import time

import SDL_Pi_HDC1080

Main Program

print

```
print ("")

print ("Read Temperature and Humidity from HDC1080 using I2C bus ")

print ("")

hdc1080 = SDL_Pi_HDC1080.SDL_Pi_HDC1080()

while True:

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

print ("Temperature = %3.1f C" % hdc1080.readTemperature())

print ("Humidity = %3.1f %%" % hdc1080.readHumidity())

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

time.sleep(3.0)
```

GITHUB, REPOZYTORIUM DOBRYCH RZECZY

Github.com to internetowa usługa hostingowa do kontroli wersji przy użyciu Git, dobrze znanego systemu zapewniającego kontrolę źródła oprogramowania. Jest używany głównie do kodu komputerowego. Zapewnia kontrolę dostępu i funkcje współpracy, takie jak śledzenie błędów, żądania funkcji, zarządzanie zadaniami i inne usługi dla każdego projektu. Od czerwca 2018 r. GitHub ma ponad 28 milionów użytkowników i 57 milionów repozytoriów, co czyni go największym hostem kodu źródłowego na świecie.

W 2018 roku GitHub został przejęty przez Microsoft, Inc., który zobowiązał się do umożliwienia github.com działania jako niezależnego oddziału. Jak na razie dobrze.

4. Uruchom kod, wpisując:

sudo python3 temperatureTest.py

Powinieneś zobaczyć następujący wynik, z nowymi odczytami temperatury i wilgotności co trzy sekundy:

```
Read Temperature and Humidity from HDC1080 using I2C bus

------------------

Temperature = 24.2 C

Humidity = 32.9 %

------------------

------------------

Temperature = 24.2 C

Humidity = 32.9 %

------------------

------------------

Temperature = 24.2 C
```

Humidity = 32.9 %

Odczytujesz teraz dane środowiskowe z urządzenia I2C. Twoje Raspberry Pi jest połączone z prawdziwym światem.

Spróbuj tego eksperymentu. Dmuchnij w płytkę czujnika HDC1080 i obserwuj, jak rośnie wilgotność! Zobaczysz coś takiego:

Temperature = 24.2 C

Humidity = 32.9 %

Temperature = 24.1 C

Humidity = 33.6 %

Temperature = 24.1 C

Humidity = 33.9 %

Temperature = 24.1 C

Humidity = 36.3 %

Temperature = 24.1 C

Humidity = 36.5 %

Rozbicie programu

Pierwsza linia importuje bibliotekę Python sys:

import sys

Następna linia mówi Pythonowi, aby przeszukał katalog SDL_Pi_HDC1080_Python3 poniżej naszego bieżącego katalogu, aby mógł znaleźć naszą bibliotekę:

sys.path.append('./SDL_Pi_HDC1080_Python3')

Więcej importu:

import time

import SDL_Pi_HDC1080

Ta instrukcja tworzy instancję obiektu hdc1080 i inicjuje go:

Main Program

print

print ("")

print ("Read Temperature and Humidity from HDC1080 using I2C bus ")

print ("")

hdc1080 = SDL_Pi_HDC1080.SDL_Pi_HDC1080()

Zestawienia te odczytują temperaturę i wilgotność i drukują je w oknie terminala. Uwaga: Widzisz, że cała złożoność korzystania z urządzenia I2C jest ukryta za pomocą biblioteki HDC1080:

while True:

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

print ("Temperature = %3.1f C" % hdc1080.readTemperature())

print ("Humidity = %3.1f %%" % hdc1080.readHumidity())

Śpij przez trzy sekundy, a następnie powtórz:

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

time.sleep(3.0)

Teraz, gdy masz ten program, możesz dodać do niego różne rzeczy, takie jak włączenie czerwonej diody LED, jeśli zrobi się za gorąco, lub włączenie niebieskiej diody LED, jeśli zrobi się zimno. Możesz nawet tweetować swoją temperaturę i wilgotność, korzystając z https://python-twitter.readthedocs.io biblioteki Pythona.

PATRZĘ NA STEROWNIK I2C

Urządzenia I2C mają adres (jak 0x40 adres naszego HDC1080) i mają również rejestry. Można o nich myśleć jako o numerowanych wskaźnikach, dla których należy pisać polecenia i odczytywać dane. HDC1080 ma osiem różnych rejestrów z różnymi adresami szesnastkowymi, jak pokazano na poniższym rysunku.

Pointer	Name	Reset value	Description
0x00	Temperature	0x0000	Temperature measurement output
0x01	Humidity	0x0000	Relative Humidity measurement output
0x02	Configuration	0x1000	HDC1080 configuration and status
0xFB	Serial ID	device dependent	First 2 bytes of the serial ID of the part
0xFC	Serial ID	device dependent	Mid 2 bytes of the serial ID of the part
0xFD	Serial ID	device dependent	Last byte bit of the serial ID of the part
0xFE	Manufacturer ID	0x5449	ID of Texas Instruments
0xFF	Device ID	0x1050	ID of the device

Sterownik I2C zasadniczo odczytuje i zapisuje z tych adresów, aby sterować HDC1080 oraz odczytywać dane dotyczące temperatury i wilgotności. Poniższy rysunek pokazuje format rejestru temperatury znajdującego się pod adresem wskaźnika 0x00.

Name	Bits		Description
TEMPERATURE	[15:02]	Temperature	Temperature measurement (read only)
	[01:00]	Reserved	Reserved, always 0 (read only)

Z biblioteki Pythona SDL_Pi_HDC1080 spójrzmy na kod Pythona, aby faktycznie odczytać ten rejestr I2C:

```
def readTemperature(self):

s = [HDC1080_TEMPERATURE_REGISTER] # temp

s2 = bytearray( s )

HDC1080_fw.write( s2 )

time.sleep(0.0625) # From the data sheet

#read 2 byte temperature data

data = HDC1080_fr.read(2)

buf = array.array('B', data)

# Convert the data

temp = (buf[0] * 256) buf[1]

cTemp = (temp / 65536.0) * 165.0 - 40

return cTemp
```

Wygląda to o wiele bardziej przerażająco niż jest w rzeczywistości. Rozkładając to, najpierw definiujemy funkcję:

```
def readTemperature(self):
```

Formatujemy adres wskaźnika (w tym przypadku 0x00) w tablicę bajtów:

```
s = [HDC1080_TEMPERATURE_REGISTER] # temp

s2 = bytearray( s )
```

Ustawiamy odczyt z rejestru wskaźników:

HDC1080_fw.write(s2)

Mamy opóźnienia 6,24 ms, zgodnie z wymaganiami arkusza danych:

time.sleep(0.0625) # Z arkusza danych

Odczytujemy dwa bajty:

#odczytaj 2-bajtowe dane temperatury

data = HDC1080_fr.read(2)

I umieść dwa bajty w tablicy bajtów:

buf = array.array('B', data)

Następnie konwertujemy bajty danych za pomocą formuły w arkuszu danych:

Konwertuj dane

temp = (buf[0] * 256) buf[1]

cTemp = (temp / 65536,0) * 165,0 – 40

I wyślij temperaturę z powrotem do programu wywołującego:

return cTemp

Istnieje wiele różnych sterowników i programów niskiego poziomu do odczytu i zapisu z urządzeń I2C na Raspberry Pi. To jest przykład jednej z najczęstszych metod. Inne metody obejmują Adafruit_i2c, SMBUS, PyComms, Quick2Wire i inne. Zwykle korzystamy z biblioteki SMBUS, ale od czasu do czasu natkniesz się na urządzenie, które do działania wymaga pewnych funkcji innych niż SMBUS.

Zabawny eksperyment polegający na pomiarze tlenu i płomienia

W tym bardziej złożonym eksperymencie bierzemy czujnik tlenu Grove i umieszczamy go pod mniej lub bardziej szczelnym szklanym słojem z zapaloną świecą. Chodzi o to, aby zmierzyć tlen w szklanym słoju i obserwować, jak poziom spada, gdy świeca zużywa tlen. Po szybkim przeszukaniu Internetu spodziewamy się, że zanim płomień zgaśnie, spadnie on o około 30 procent. Byłoby to od 21 procent tlenu do około 14,7 procent tlenu. Przechowujemy informacje w pliku CSV (plik rozdzielony przecinkami) do późniejszego wykreślenia za pomocą MatPlotLib. Możesz również łatwo odczytać te dane w arkuszu kalkulacyjnym Excel i wykreślić je za pomocą Excela. MatPlotLib to biblioteka Pythona do tworzenia wykresów jakości publikacji przy użyciu metod podobnych do MATLIB. Możesz wyprowadzać formaty, takie jak PDF, Postscript, SVG i PNG. Następnie zapaliliśmy świeczkę i obserwowaliśmy dane w oknie połączonej przeglądarki

Raspberry Pi. Co potrzebujemy do wykonania tego eksperymentu:

Przetwornik analogowo-cyfrowy: konwertuje wyjście analogowe czujnika tlenu na dane cyfrowe dla Raspberry Pi.

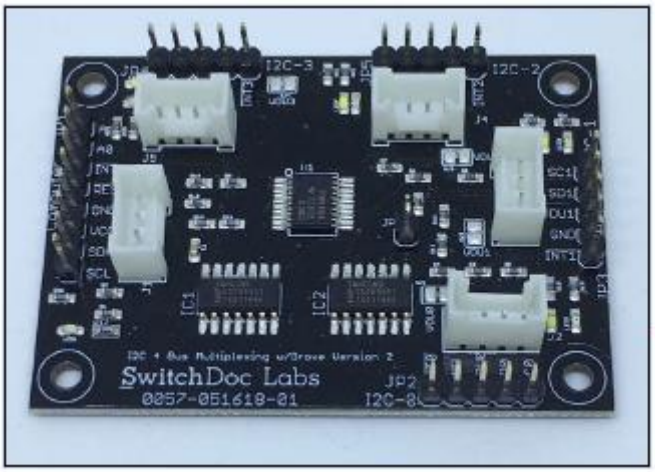
Czujnik tlenu Grove: Ten czujnik mierzy procent tlenu w powietrzu i przetwarza go na wartość analogową (0 – 5 V).

Świeca: Świeca zużyje tlen w misce. Możesz użyć dowolnej świecy, o ile znajduje się ona pod misą.

Duża szklana miska: miska zakryje i uszczelni świecę, aby zmierzyć tlen

Przetworniki analogowo-cyfrowe (ADC)

Przetwornik analogowo-cyfrowy pobiera sygnał analogowy (zobacz różnicę między sygnałem analogowym i cyfrowym w rozdziale 2 tego minibooka) i konwertuje go na sygnał cyfrowy (w tym przypadku 16 bitów), który komputer może odczytać . Gdy masz cyfrową liczbę w komputerze, możesz przeskalować ją z powrotem do woltów, mnożąc ją przez (5,0/65535,0), aby uzyskać liczbę zmiennoprzecinkową reprezentującą wolty. Nie ma co do tego wątpliwości. Brak przetwornika analogowo-cyfrowego to prawdziwy cios dla Raspberry Pi. Przetwornik analogowo-cyfrowy Grove, którego używamy w tym eksperymencie, to czterokanałowy, 16-bitowy przetwornik analogowo-cyfrowy Grove dostępny na store.switchdoc. com, a także na Amazon.com. (Patrz rysunek 3-5.)



Istnieją inne moduły Grove ADC dostępne na eeedstudio.com, ale chcieliśmy użyć 16-bitowego przetwornika ADC dla większej dokładności i faktu, że ma cztery kanały zamiast tylko jednego kanału

Czujnik tlenu Grove

Czujnik gazu Grove (O2) to czujnik do badania stężenia tlenu w powietrzu. (Patrz rysunek 3-6.)

Wykrywa aktualne stężenie tlenu i wysyła wartości napięcia proporcjonalne do stężenia tlenu. Liczby te można zinterpretować, odwołując się do wykresu charakterystyki liniowej stężenia tlenu. Ta wartość czujnika odzwierciedla jedynie przybliżony trend stężenia gazowego tlenu w dopuszczalnym zakresie błędu, nie reprezentuje dokładnego stężenia gazowego tlenu. Wykrywanie pewnych składników w powietrzu zwykle wymaga bardziej precyzyjnego i kosztownego przyrządu, czego nie można wykonać za pomocą pojedynczego czujnika gazu. Ten czujnik wymaga również około 30-minutowego czasu nagrzewania.

Podłączanie eksperymentu z tlenem

Do tej pory masz już spore doświadczenie w podłączaniu urządzeń Grove do Raspberry Pi. Wykonaj następujące kroki, aby skonfigurować czujnik tlenu:

1. Odłącz zasilanie od Raspberry Pi.

2. Podłącz kabel Grove do czujnika tlenu Grove, a następnie do złącza Grove oznaczonego A1 na czterokanałowej, 16-bitowej płytce ADC Grove.

3. Podłącz kolejny kabel Grove do złącza Grove oznaczonego I2C na czterokanałowej, 16-bitowej karcie ADC Grove. Podłącz drugi koniec tego kabla Grove do jednego ze złączy oznaczonych jako I2C na płycie Pi2Grover podłączonej do Raspberry Pi. (Patrz rysunek 3-7.)

4. Podłącz zasilanie do Raspberry Pi.

5. Uruchom polecenie i2cdetect -y 1 w oknie terminala. Powinieneś zobaczyć to wyjście:

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Adres 0x48 to czterokanałowa, 16-bitowa płytka ADC Grove. Jeśli tego nie widzisz, wróć i sprawdź okablowanie. Teraz przetestujesz konfigurację, uruchamiając prosty program w języku Python. Najpierw utwórz nowy katalog dla programu:

cd

mkdir oxygenProject

cd oxygenProject

git clone https://github.com/switchdoclabs/SDL_Pi_Grove4Ch16BitADC

Następnie wprowadź następujący kod do pliku o nazwie senseOxygen.py w oknie terminala za pomocą nano:

```
import time, sys

sys.path.append('./SDL_Pi_Grove4Ch16BitADC/SDL_Adafruit_ADS1x15')

import SDL_Adafruit_ADS1x15

ADS1115 = 0x01 # 16-bit ADC

# Select the gain

gain = 6144 # /- 6.144V

# Select the sample rate

sps = 250 # 250 samples per second

# Initialize the ADC using the default mode (use default I2C address)

adc = SDL_Adafruit_ADS1x15.ADS1x15(ic=ADS1115)

dataFile = open("oxygenData.csv",'w')

totalSeconds = 0

while (1):

# Read oxygen channel in single-ended mode using the settings above

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

voltsCh1 = adc.readADCSingleEnded(1, gain, sps) / 1000

rawCh1 = adc.readRaw(1, gain, sps)

# O2 Sensor

sensorVoltage = voltsCh1 *(5.0/6.144)

AMP = 121

K_O2 = 7.43

sensorVoltage = sensorVoltage/AMP*10000.0

Value_O2 = sensorVoltage/K_O2 - 1.05

print ("Channel 1 =%.6fV raw=0x%4X O2 Percent=%.2f" % (voltsCh1, rawCh1,

Value_O2 ))

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

dataFile.write("%d,%.2f\n" % (totalSeconds, Value_O2))

totalSeconds = totalSeconds 1
```

dataFile.flush()

time.sleep(1.0)

Kiedy skończysz używać czujnika tlenu, upewnij się, że umieściłeś go z powrotem w dołączonym zamkniętym pojemniku i zamknij górną część. Wilgoć z czasem zniszczy czujnik. (W przeszłości zniszczyliśmy te czujniki.) Po uruchomieniu programu, oto wyniki:

Channel 1 =2.436375V raw=0x32C2 O2 Percent= 22.05

Channel 1 =2.436375V raw=0x32C2 O2 Percent= 22.05

Channel 1 =2.436375V raw=0x32C1 O2 Percent= 22.05

Channel 1 =2.436375V raw=0x32C2 O2 Percent= 22.05

Channel 1 =2.436187V raw=0x32C1 O2 Percent= 22.05

Łamanie kodu

W tych instrukcjach ustawiamy parametry modułu ADC:

import time, sys

sys.path.append('./SDL_Pi_Grove4Ch16BitADC/SDL_Adafruit_ADS1x15')

import SDL_Adafruit_ADS1x15

Normal Imports. Notice the path goes to the subdirectory in the your directory.

ADS1115 = 0x01 # 16-bit ADC

Select the gain

gain = 6144 # /- 6.144V

Select the sample rate

sps = 250 # 250 samples per second

Następnie otwieramy plik tekstowy, w którym przechowujemy nasze dane, które można później sporządzić na wykresie w programie Excel lub w inny sposób:

```
# Initialize the ADC using the default mode (use default I2C address)

adc = SDL_Adafruit_ADS1x15.ADS1x15(ic=ADS1115)

dataFile = open("oxygenData.csv",'w')
```

Odczytaj dane z ADC. Wolty i surowe dane (surowe dane tylko w celach informacyjnych):

```
totalSeconds = 0

while (1):

# Read oxygen channel in single-ended mode using the settings above

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

voltsCh1 = adc.readADCSingleEnded(1, gain, sps) / 1000

rawCh1 = adc.readRaw(1, gain, sps)
```

To jest ze specyfikacji czujnika O2, jak obliczyć procent O2 z napięcia z ADC:

```
# O2 Sensor

sensorVoltage = voltsCh1 *(5.0/6.144)

AMP = 121

K_O2 = 7.43

sensorVoltage = sensorVoltage/AMP*10000.0

Value_O2 = sensorVoltage/K_O2 - 1.05
```

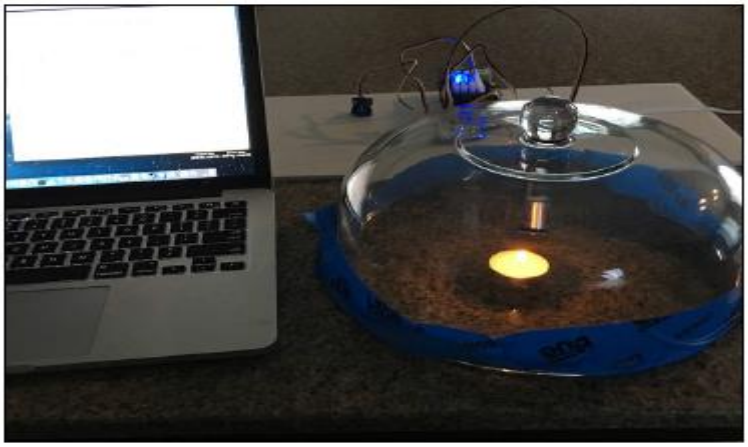
Tutaj zapisujesz dane do pliku:

```
print ("Channel 1 =%.6fV raw=0x%4X O2 Percent=%.2f" % (voltsCh1, rawCh1,

Value_O2 ))

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

dataFile.write("%d,%.2f\n" % (totalSeconds, Value_O2))
```

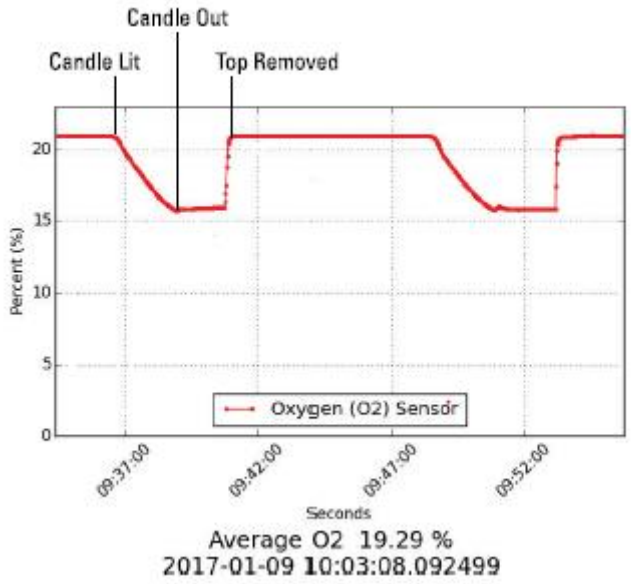
Psujemy plik, aby upewnić się, że ostatnia wartość jest zapisywana w pliku. W końcu zakończysz ten program za pomocą Ctrl-C:

```
totalSeconds = totalSeconds 1

dataFile.flush()
```

Teraz, gdy masz już zbudowane całe oprogramowanie, weź świecę, umieść ją pod miską z czujnikiem tlenu, uruchom program i zapal świecę.



Po chwili płomień zgaśnie. Zatrzymaj swój program za pomocą Ctrl-C i spójrz na swoje dane i wykreśl je.



Patrząc na liczby, ustaliliśmy, że zaczęliśmy od około 21 procent tlenu, a świeca zgasła przy około 15,8 procentach tlenu, co oznacza redukcję o około 25 procent. To mniej niż oczekiwane 30-procentowe zmniejszenie poziomu tlenu. Różnica? Domyślałbym się kombinacji dokładności czujnika i rodzaju świecy.

Jeszcze jedna rzecz do zapamiętania: spójrz na wykres zaraz po zgaśnięciu świecy. Widać, że uszczelnienie nie było idealne, ponieważ tlen zaczął się skradać.

Tworzenie pulpitu nawigacyjnego w telefonie za pomocą Blynk i Python

Kiedy prowadzisz samochód, wszystkie informacje o tym, jak szybko jedziesz, ile pozostało paliwa i inne informacje o samochodzie, znajdują się na desce rozdzielczej. Pokażemy Ci, jak zbudować prosty pulpit nawigacyjny, dzięki któremu będziesz mógł przeglądać dane swojego projektu na swoim smartfonie. Aby zilustrować, jak to zrobić, użyjemy bezpłatnej aplikacji Blynk (bezpłatnej dla małych pulpitów nawigacyjnych; pobierają trochę opłaty za więcej energii do zbudowania większej liczby elementów

sterujących). Ta aplikacja jest dostępna w różnych sklepach z aplikacjami na Androida i iPhone'a. Użyjemy iPhone'a, aby pokazać użycie, ale jest prawie identyczny dla telefonów z Androidem.

Redux czujnika temperatury i wilgotności HDC1080

Wcześniej w tym rozdziale zbudowałeś projekt czujnika temperatury i wilgotności za pomocą Raspberry Pi. Chwyć ten projekt teraz i napiszmy trochę więcej oprogramowania, aby połączyć go z Blynk i wyświetlić nasze wartości na urządzeniu. Tutaj rysunek 3.10 pokazuje nam, jak będzie wyglądał nasz pulpit nawigacyjny.

INNE DESKI ROZDZIELCZE

Blynk nie jest jedynym internetowym pulpitem nawigacyjnym. Możesz także sprawdzić:

- Darmowa deska

- XOBXOB

- Adafruit IO

- ThinkSpeak

— Chmura IBM

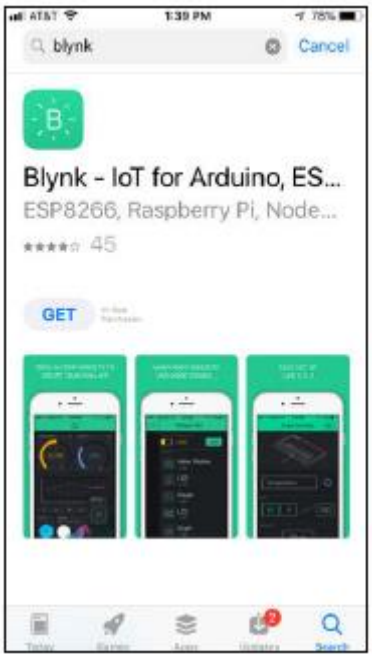
- Stan początkowy

Wszystkie mają różne mocne i słabe strony. Wszystkie mają jakąś darmową opcję, która regularnie się zmienia.

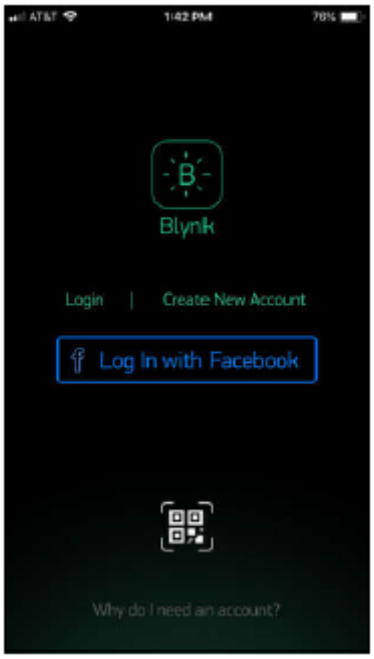
Jak dodać pulpit nawigacyjny Blynk

Najpierw pokażemy, jak skonfigurować aplikację Blynk. Odbywa się to na iPhonie, ale jest bardzo podobne do używania go na telefonie z Androidem. A Python jest identyczny w obu przypadkach!

1. Zainstaluj aplikację Blynk na swoim telefonie komórkowym



2. Otwórz aplikację Blynk i utwórz konto. Musisz podać konto i adres e-mail, ale nie pobierają za to żadnych opłat.



3. Kliknij przycisk, aby zeskanować kod QR.

4. Zeskanuj kod QR pokazany na rysunku.

5. Na ekranie zobaczysz teraz aplikację MyTemperature.

6. Kliknij środek projektu, aby wybrać projekt. Następnie kliknij wskazany przycisk, aby przejść do ustawień projektu.

Uwaga: Teraz skopiuj i wklej token uwierzytelniania (TOKEN AUTH) do wiadomości e-mail do siebie lub do innego bezpiecznego dokumentu, ponieważ umieścimy go w pliku programu Python temperatureTest.py w następnej sekcji. Rysunek 3-16 przedstawia ekran początkowy aplikacji Blynk. Rysunek przedstawia kod uwierzytelniający.

Masz teraz załadowaną aplikację myTemperature. Instalacja aplikacji Blynk myTemperature została zakończona. Teraz zmodyfikujmy oprogramowanie, aby obsługiwało aplikację Blynk.

Zmodyfikowane oprogramowanie temperatureTest.py dla aplikacji Blynk

Aby zmodyfikować oprogramowanie do obsługi aplikacji Blynk, wykonaj następujące kroki:

1. Utwórz katalog w głównym katalogu, wprowadzając następujące informacje:

cd

mkdir myTemperature

cd myTemperature

Teraz jesteś w katalogu myTemperature.

2. Zanim spojrzysz na kod Pythona do odczytu, a następnie „Blynking" swojej temperatury, zainstaluj bibliotekę na Raspberry Pi. Robisz to, „klonując" bibliotekę znajdującą się na github.com, używając następującego polecenia w oknie terminala:

git clone https://github.com/switchdoclabs/SDL_Pi_HDC1080_Python3.git

3. Wprowadź poniższy kod do pliku o nazwie myTemperature.py za pomocą nano lub swojego ulubionego edytora.

```
#!/usr/bin/env python3

#imports

import sys

sys.path.append('./SDL_Pi_HDC1080_Python3')

import time

import SDL_Pi_HDC1080

import requests

import json

BLYNK_URL = 'http://blynk-cloud.com/'

BLYNK_AUTH = 'xxxx'

# Main Program

print

print ("")

print ("Read Temperature and Humidity from HDC1080 using I2C bus and send

to Blynk ")

print ("")

hdc1080 = SDL_Pi_HDC1080.SDL_Pi_HDC1080()

def blynkUpdate(temperature, humidity):

print ("Updating Blynk")

try:
```

```
put_header={"Content-Type": "application/json"}
val = temperature
put_body = json.dumps(["{0:0.1f}".format(val)])
r = requests.put(BLYNK_URL BLYNK_AUTH '/update/V0', data=put_body,
headers=put_header)
put_header={"Content-Type": "application/json"}
val = humidity
put_body = json.dumps(["{0:0.1f}".format(val)])
r = requests.put(BLYNK_URL BLYNK_AUTH '/update/V1', data=put_body,
headers=put_header)
put_header={"Content-Type": "application/json"}
val = time.strftime("%Y-%m-%d %H:%M:%S")
put_body = json.dumps([val])
r = requests.put(BLYNK_URL BLYNK_AUTH '/update/V2', data=put_body,
headers=put_header)
return 1
except Exception as e:
print ("exception in updateBlynk")
print (e)
return 0
while True:
temperature = hdc1080.readTemperature()
humidity = hdc1080.readHumidity()
print ("------------------")
print ("Temperature = %3.1f C" % hdc1080.readTemperature())
print ("Humidity = %3.1f %%" % hdc1080.readHumidity())
print ("------------------")
blynkUpdate(temperature, humidity)
time.sleep(3.0)
```

Ten kod aktualizuje Twoją aplikację Blynk co trzy sekundy. Jeśli aktualizujesz aplikację Blynk częściej niż raz na sekundę, możesz zostać rozłączony z serwerem. Staraj się, aby Twoje żądania wysyłały mniej niż dziesięć wartości na sekundę, aby być dobrym obywatelem Blynka.

4. Ostatnią rzeczą, którą musisz zrobić przed uruchomieniem kodu, jest zastąpienie „xxxx” kodem autoryzacyjnym Blynk, który będzie wyglądał mniej więcej tak: 445730794c1c4c8ea7852a31555f44444.

Zanim:

BLYNK_AUTH = 'xxxx'

Po:

BLYNK_AUTH = '445730794c1c4c8ea7852a31555f44444'

Uwaga: musisz użyć kodu autoryzacyjnego otrzymanego e-mailem (lub wyciąć i wkleić z aplikacji), w przeciwnym razie nie połączysz się z aplikacją. Pokazany tutaj przykładowy kod nie zadziała.

Łamanie kodu

Ten kod jest bardzo podobny do kodu HDC1080 z wcześniejszej części tego rozdziału, z wyjątkiem kodu blynkUpdate.

def blynkUpdate(temperature, humidity):

print ("Updating Blynk")

try:

Dlaczego mamy tutaj „próbę”? Ponieważ czasami biblioteka żądań zgłosi błąd, jeśli Internet nie działa. Następnie ustawiamy wymagany nagłówek http dla biblioteki żądań:

put_header={"Content-Type": "application/json"}

BIBLIOTEKA ŻĄDAŃ

Biblioteka żądań Pythona jest jedną z najbardziej przydatnych bibliotek do komunikacji przez Internet przy użyciu żądań http. Jest przeznaczony do użytku przez ludzi do interakcji z żądaniami HTTP bez ujawniania złożoności żądań. Bardzo Pythoniczny. Biblioteka żądań umożliwia wysyłanie żądań HTTP/1.1 przy użyciu języka Python. Umożliwia także dostęp do danych odpowiedzi na żądania za pomocą Pythona.

Poniższy kod ustawia liczbę cyfr po prawej stronie przecinka dziesiętnego na 1, dzięki czemu nie będziemy mieć długich liczb stosunkowo bezsensownych cyfr ze względu na dokładność HDC1080:

val = temperature

put_body = json.dumps(["{0:0.1f}".format(val)])

Poniższy kod wykonuje rzeczywiste przesyłanie danych do serwera Blynk w formie żądania http:

r = request.put(BLYNK_URL BLYNK_AUTH '/update/V0', data=put_body,

headers=put_header)

Tutaj drukujemy każdy wyjątek na ekranie terminala. Pomaga to również ustalić, czy kod autoryzacyjny Blynk został ustawiony nieprawidłowo:

oprócz Wyjątku jako e:

except Exception as e:

```
print ("exception in updateBlynk")

print (e)

return 0
```

Następnie uruchommy program:

Sudo python3 myTemperature.py

Zobaczysz ten typ danych wyjściowych na ekranie terminala:

```
------------------

Temperatura = 22,6 C

Wilgotność = 36,8%

------------------

Aktualizowanie Blynka

------------------

Temperatura = 22,5 C

Wilgotność = 36,8%

------------------

Aktualizowanie Blynka
```

Naciśnij klawisz Run w prawym górnym rogu aplikacji Blynk na telefonie, a następnie obserwuj, jak zaczynają napływać dane. Jeśli nie otrzymasz danych w ciągu kilku sekund, sprawdź swój kod uwierzytelniający i upewnij się, że zacząłeś aplikację, naciskając przycisk Start w prawym górnym rogu aplikacji. Twoje wyniki powinny wyglądać jak na rysunku 3

Zwróć uwagę, że wyświetlanie na żywo na wykresie może wyglądać trochę dziwnie, ale inne ekrany zaczną się wypełniać i wyglądać naprawdę dobrze.

Dokąd się stąd udać

Nauczyłeś się wiele o tym, jak połączyć się z prawdziwym Pythonem na swoim Raspberry Pi. Proponujemy inne ciekawe rzeczy do zrobienia, opierając się na nowej wiedzy specjalistycznej:

Dodaj więcej czujników I2C do swojego Raspberry Pi. Są ich setki.

Spróbuj dodać kilka czujników cyfrowych do swojego Pi, takich jak detektory PIR, aby wykrywać ciepłe ciała, takie jak ludzie, przed twoim Pi.

Dodaj kompas I2C i akcelerometr do swojego Pi.

Twórz większe i bardziej złożone pulpity nawigacyjne i pokazuj swój projekt znajomemu (tak, możesz to zrobić, udostępniając znajomym swój kod uwierzytelniający).

Dodaj silnik, aby rzeczy się poruszały.