Dozownik pokarmu z Raspberry Pi

Komputerki jednopłytkowe świetnie nadają się do automatyzacji różnorodnych zadań. Choć czasem może się wydawać, że użycie komputera do realizacji łatwej czynności, takiej jak np. karmienie zwierząt, jest „przerostem formy nad treścią”. Zdarzają się sytuacje, gdy projekty tego typu pozwalają rozwiązać ważniejsze, realne, codzienne problemy. Ponadto, mogą stanowić podstawę do wykonania bardziej zaawansowanych, użytecznych w przemyśle rozwiązań, czego dobrym przykładem jest opisywany poniżej karmnik dla kotów.

Potrzebne podzespoły:
- Pojemnik na płatki śniadaniowe z dozownikiem.
- Raspberry Pi,
- 2 serwomechanizmy standardowe.
- Opcjonalnie – karta sieciowa Wi-Fi z interfejsem USB.
- Uniwersalna płytka drukowana.
- Goldpiny do wygodnego przyłączenia serwomechanizmów.
- Przewody do wykonania połączeń.
- 4 oporniki o rezystancji 380 Ω.
- 10-watowy zasilacz +5 V z wyjściem USB.
- 2 przełączniki z opcjonalnymi diodami LED.
- 2 dodatkowe diody LED.
- Buzzer piezoelektryczny.
- 2 rury PVC jako do kierowania pokarmu.
- 8 plastikowych opasek zaciskowych.
- Taśma montażowa.
- Rurka termokurczliwa.

Samodzielne karmienie zwierząt domowych to przyjemna czynność, którą lubi większość właścicieli zwierzaków. Niemniej bywa też tak, że właściciel musi opuścić mieszkanie na pewien czas i nie ma z kim zostawić swoich podopiecznych ani też kogo poprosić o ich karmienie. Oczywiście, można po prostu spróbować nałożyć zwierzętom do misek kilka czy kilkanaście porcji na raz, ale nierzadko „znikną” one zbyt szybko, a przez pozostałe dni koty będą głodować. Problemem jest też sytuacja, gdy podanie zbyt dużej ilości pokarmu nie jest możliwe z innych względów – np. gdy nie ma wystarczającej ilości miejsca w klatce czy też, gdy zbytnio zmieni on swoje właściwości (choćby konsystencję), po tym, gdy trafi choćby do akwarium z wodą.

Zaawansowane możliwości

Skomputeryzowany dozownik nie tylko automatyzuje podawanie pokarmu, ale w wersji rozwiniętej może też wykonywać dodatkowe funkcje. Po pierwsze może modyfikować ilość udostępnianego jedzenia w zależności od sygnałów z czujników. Może też podawać różne jedzenie np. w oparciu o kalendarz, by utrzymywać zwierzęciu odpowiednią dietę. Ponieważ Raspberry Pi jest wyposażone w interfejsy ethernetowe, komputerem można też ewentualnie sterować zdalnie przez Internet, samodzielnie – czy to na wakacjach, czy w delegacji – dobierając ilość pokarmu do odczytów z czujników. Przedstawiony projekt, wykonany przez Davida Bryana i opublikowany na jego blogu, nie realizuje jednak tak zaawansowanych funkcji, ale stanowi dobrą bazę do rozwoju.

Źródło projektu
Autorem omawianego projektu jest David Bryan, a komplet zdjęć, kod źródłowy i anglojęzyczny opis można znaleźć pod adresem: http://goo.gl/BgGrjS

Potrzebne elementy i narzędzia

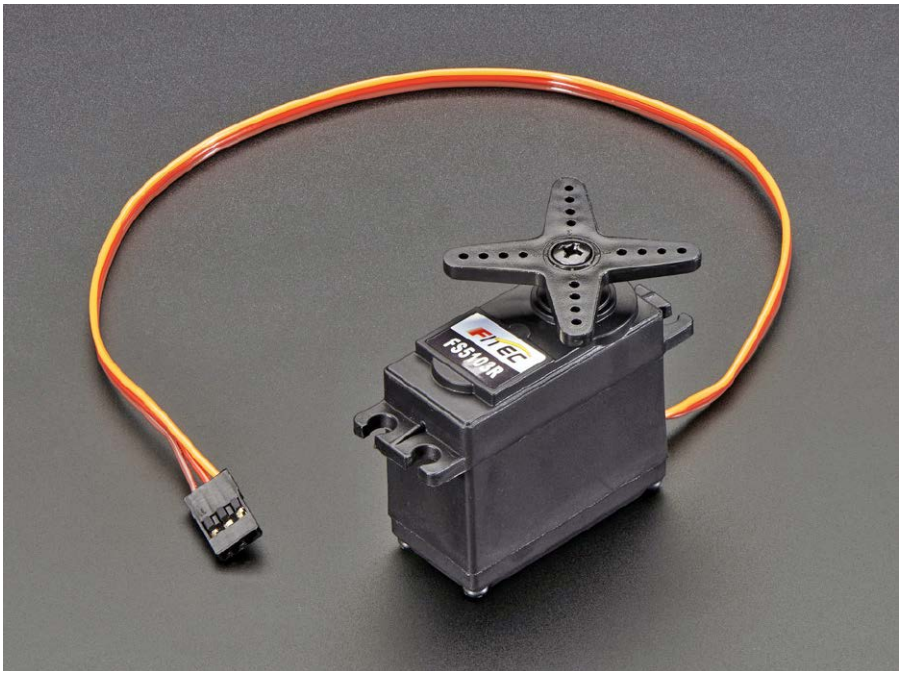
Podstawowym elementem projektu jest plastikowy dozownik do przekąsek i płatków śniadaniowych, jaki można spotkać np. w hotelach. Jego konkretny kształt nie jest istotny – ważne tylko, by miał obrotowe łopatki dozujące, do których podłączony zostanie serwomechanizm. W zależności od fizycznego ustawienia dozownika, przydatne mogą okazać się rurki prowadzące pokarm do misek. Można w tym celu użyć tanich rur PCV, dostępnych w dowolnym sklepie z wyposażeniem domu i ogrodu. Autor projektu połączył całość za pomocą taśmy montażowej i plastikowych opasek zaciskowych (niekiedy nazywanych w Polsce „trytytkami”) oraz skorzystał z rurek termokurczliwych do izolacji elementów elektronicznych. Opcjonalnym wyposażeniem będzie buzzer i diody LED, sygnalizujące podanie pokarmu, by przywołać zwierzęta. Można też zainstalować przyciski, którymi właściciel będzie mógł ręcznie sterować dozownikiem, bez potrzeby korzystania z interfejsu sieciowego.

Naturalnie konieczny będzie też sam komputer – w tym przypadku Raspberry Pi, a ponadto by uniknąć potrzeby prowadzenia przewodu sieciowego, autor dokupił kartę sieciową Wi-Fi na USB. Nie obyło się też bez kilku dodatkowych elementów elektronicznych: płytki drukowanej ułatwiającej łączenie elementów z Raspberry Pi, goldpinów, przewodów, rezystorów i zasilacza. Łączny koszt podzespołów wyniósł niemal 150 dolarów, z czego najdroższy był sam podwójny dyspenser przekąsek, a w drugiej kolejności był to Raspberry Pi oraz dwa serwomechanizmy. Autor zastosował też dosyć drogie przełączniki z dodatkowymi diodami LED i zapłacił 8,5 dolara za przewody, co podniosło łączna kwotę.

Narzędzia potrzebne do wykonania projektu to przede wszystkim lutownica i miniwiertarka. Przydatny będzie też pistolet klejowy oraz oczywiście szczypce i obcążki do przecinania przewodów. Autor używał dwóch wierteł o średnicach 0,5” i 1”.

Montaż

Wykonanie projektu wymaga nieco pracy ręcznej i wprawy w posługiwaniu się narzędziami, ale szczegóły techniczne zależą od wybranej konfiguracji i cech fizycznych nabytego dozownika. W praktyce najczęściej konieczne jest zdjęcie lub odcięcie uchwytów, za pomocą których obracane są łopatki dozownika. W pozostałych po nich, wystających elementach należy wykonać nacięcia (nawiercenia), tak by można je było połączyć z serwomechanizmami. Ważne, by połączenie było jak najdokładniej wykonane – oś serwomechanizmu powinna pokrywać się z osią łopatek dozownika. Warto też, ale tylko dla likwidacji ewentualnych luzów przy jednoczesnym wprowadzeniu pewnej elastyczności połączenia, zastosować klej termiczny. Trzeba także sprawdzić, czy siła potrzebna do obrotu łopatek



Fotografia 1. Serwomechanizmy zastosowany w projekcie

z załadowanym w dozowniku pokarmem nie jest zbyt duża i dobrać odpowiednie serwomechanizmy. Autor użył serwomechanizmów FeeTech FS5103R (**fotografia 1**), zasilanych napięciem 5 V o momencie siły przekraczającym 0,3 Nm i wymiarach 37 mm×54 mm×20 mm. Przypuszczalnie trzeba w nich wyłamać ogranicznik obrotu osi, ponieważ serwomechanizmy będą napędzały łopatki śmigiełka dozownika, które musi obracać się dookoła. Plastikowe opaski zaciskowe mogą posłużyć do przytwierdzenia obudów serwomechanizmów do obudowy dozownika, a do ujść dozownika, za pomocą taśmy montażowej mocujemy rury prowadzące do misek.

Połączenia elektryczne ograniczamy do minimum. Serwomechanizmy zasilamy napięciem 5 V, czyli takim samym jak samo Raspberry Pi. Naturalnie przyłączamy je także do masy, a trzeci przewód dołączamy do wyprowadzeń GPIO Raspberry Pi. W omawianym projekcie autor skorzystał z wyprowadzeń o numerach 18 i 23, korzystając z nomenklatury wyprowadzeń układu BCM2835, do której następnie odnosi się w programie. Podłączamy opcjonalne diody poprzez rezystory, buzzer (do masy i pinu #24) i przełączniki. Lista połączeń znalazła się w **tabeli 1**. Jeśli dobrze dobierzemy dozownik, całość elektroniki zmieści się w podstawce obudowy.

Potrzebne oprogramowanie

Do Raspberry Pi wkładamy kartę SD z zainstalowanym na niej systemem operacyjnym. System wgrywamy z obrazu z Internetu – np. Raspbian, który wymaga karty o pojemności przynajmniej 4 GB. Kod programu został przygotowany w języku Python, którego obsługa jest zaimplementowana w Raspbianie. Potrzebne są natomiast dodatkowe biblioteki Pythona, do których instalacji przydatny będzie menedżer pakietów PIP. Instalujemy więc pakiet **python-pip**, korzystając z narzędzia instalacyjnego **apt**:

```
sudo apt-get install python-pip
```

Następnie za pomocą menedżera PIP instalujemy dodatkowe potrzebne pakiety Pythona, umożliwiające sprawdzenie strefy czasowej i określenie chwili zachodu słońca, o której automat ma wydawać pokarm. W tym celu korzystamy z poleceń:

```
sudo pip install astral
sudo pip install pytz
```

W końcu doinstalowujemy program umożliwiający programom Pythona zaawansowane sterowanie wejściami i wyjściami GPIO, a w tym generowanie sygnału PWM:

```
sudo apt-get install python-rpi.
gpio
```

Właściwy kod programu

Kod programu piszemy w pliku tekstowym i zapisujemy na dysku tak, by mógł system mógł go uruchamiać jako plik wykonywalny. Jeśli chcemy wysyłać potwierdzenia dozowania pokarmu e-mailem, tworzymy dodatkowy plik z ustawieniami, który będziemy następnie ładować w głównym pliku programu. Aby program wykonywał się automatycznie po uruchomieniu systemu operacyjnego, dodajemy go np. do listy poleceń do uruchomienia, w pliku */etc/rc.local*.

Tabela 1. Podłączenia wyprowadzeń GPIO w Raspberry PI

Nazwa GPIO	Numer wyprowadzenia	Nazwa	Oznaczenie wg BCM2835	Dołączane elementy
P1_01	1	3V3		Wspólne napięcie 3,3 V dla diod LED i przycisków
P1_02	2	5V0		Napięcie 5 V dla serwomechanizmów
P1_03	3	SDA0	GPIO0	NC
P1_04	4	DNC		NC
P1_05	5	SCL0	GPIO1	NC
P1_06	6	GND		Wspólna masa serwomechanizmów
P1_07	7	GPIO7	GPIO4	LED przycisku 1 z szeregowym opornikiem
P1_08	8	TXD	GPIO14	NC
P1_09	9	DNC		NC
P1_10	10	RXD	GPIO15	NC
P1_11	11	GPIO0	GPIO17	Przycisk 1 z rezystorem podłączonym do masy
P1_12	12	GPIO1	GPIO18	Sterowanie serwomechanizmem 1
P1_13	13	GPIO2	GPIO21	LED przycisku 2 z szeregowym opornikiem
P1_14	14	DNC		Wspólna masa serwomechanizmów i diod LED
P1_15	15	GPIO3	GPIO22	Przycisk 2 z rezystorem podłączonym do masy
P1_16	16	GPIO4	GPIO23	Sterowanie serwomechanizmem 2
P1_17	17	DNC		NC
P1_18	18	GPIO5	GPIO24	Buzzer piezoelektryczny
P1_19	19	SPI_MOSI	GPIO10	NC
P1_20	20	DNC		Wspólna masa serwomechanizmów i diod LED
P1_21	21	SPI_MISO	GPIO9	NC
P1_22	22	GPIO6	GPIO25	NC
P1_23	23	SPI_SCLK	GPIO11	NC
P1_24	24	SPI_CE0_N	GPIO8	NC
P1_25	25	DNC		Wspólna masa serwomechanizmów i diod LED
P1_26	26	SPI_CE1_N	GPIO7	NC

Kod rozpoczynamy on załadowania bibliotek:

```
import time
import smtplib
from email_settings import *
import datetime
from datetime import date,
timedelta, datetime
import RPi.GPIO as GPIO
from astral import *
from pytz import timezone
```

Pierwsza pozwala na wykonywanie operacji związanych z czasem. Druga – na wysyłanie wiadomości e-mailem. Trzecia, to utworzony przez nas plik, zawierający ustawienia konta emailowego. Czwarta i piąta pozwalają na przetwarzanie zmiennych zawierających informacje o czasie. Szósta dotyczy wspomnianej wcześniej zaawansowanej obsługi GPIO. Przedostatnia pozwala sprawdzić informacje o zachodzie słońca, a ostatnia określić strefę czasową.

Po ustawieniu informacji o strefie czasowej, można przejść do definicji funkcji oraz przypisania wyprowadzeniom GPIO nazw. Autor stworzył krótką funkcję do przesyłania wiadomości emailowych, korzystając z wcześniej wczytanych ustawień i danych serwera pocztowego, z którego miał zamiar korzystać. Definicje zmiennych związanych z wyprowadzeniami GPIO wyglądają następująco:

```
Servo1Pin=18
Servo2Pin=23
BeeperPin=24
GPIO_ButtonL_LED_PIN=4
GPIO_ButtonL_PIN=17
GPIO_ButtonR_LED_PIN=27
GPIO_ButtonR_PIN=22
```

Przy czym trzeba zaznaczyć, że wyprowadzenia GPIO #4 i #27 służą do załączania podświetlania przycisków, które autor użył w swoim projekcie. Konieczne jest też zainicjowanie wyprowadzeń, a więc przypisanie im trybów pracy. Naturalnie piny buzzera i LEDów ustawiane są jako wyjścia, a piny przycisków, jako wejścia:

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(BeeperPin, GPIO.OUT)
GPIO.setup(GPIO_ButtonL_LED_PIN,
GPIO.OUT)
GPIO.setup(GPIO_ButtonL_PIN,
GPIO.IN)
GPIO.setup(GPIO_ButtonR_LED_PIN,
GPIO.OUT)
GPIO.setup(GPIO_ButtonR_PIN,
GPIO.IN)
GPIO.setup(Servo1Pin, GPIO.OUT)
GPIO.setup(Servo2Pin, GPIO.OUT)
```

Wypada również na starcie programu określić stan początkowy wyjść, by niepotrzebnie nie zaświecać diod LED:

```
GPIO.output(BeeperPin, False)
GPIO.output(GPIO_ButtonL_LED_PIN,
False)
GPIO.output(GPIO_ButtonR_LED_PIN,
False)
```

oraz poinformować użytkownika, że urządzenie jest gotowe do pracy, emitując sygnał buzzerem:

```
GPIO.output(BeeperPin, True)
time.sleep(.25)
GPIO.output(BeeperPin, False)
time.sleep(.25)
GPIO.output(BeeperPin, True)
time.sleep(.25)
GPIO.output(BeeperPin, False)
```

Dobrym pomysłem jest stworzenie funkcji, które będą obracały serwomechanizmami przez określony czas. Ponieważ serwomechanizmy są sterowane sygnałem PWM, wyjścia oznaczone jako Servo1Pin i Servo2Pin należy w takiej funkcji odpowiednio skonfigurować, korzystając z polecenia **GPIO.PWM()**, któremu podajemy numer wyprowadzenia oraz częstotliwość sygnału PWM (wyrażoną w hercach). W przypadku użytych serwomechanizmów, poruszenie ich do przodu wymaga zastosowania impulsu o długości około 2 ms, a do tyłu, ok. 1 ms. Uzyskujemy to poprzez wygenerowanie sygnału PWM o częstotliwości 50 Hz i wypełnieniu około (odpowiednio) około 10% i 5%. Autor projektu użył wypełnień 10,5% i 4,5%, które dawały najlepsze rezultaty. Trzeba przy tym zaznaczyć, że potrzebne są oddzielne funkcje do uruchamiania jednego i dwóch serwomechanizmów jednocześnie. Wynika to faktu, że kod jest napisany synchronicznie, a w fragmencie poruszającym serwomechanizmami zastosowane jest opóźnienie, wstrzymujące działanie całego

Listing 1. Funkcje poruszające jednym i dwoma serwomechanizmami. Zgodnie z kierunkiem wskazówek zegara i w przeciwnym kierunku.

```
#jeden serwomechanizm, zgodnie ze wskazówkami zegara
def servo_CW(ServoPIN,SleepTime):
    servo = GPIO.PWM(ServoPIN, 50)
    servo.start(10.5)
    time.sleep(SleepTime)
    servo.stop()
    time.sleep(.05)

#dwa serwomechanizmy, zgodnie ze wskazówkami zegara
def dual_servo_CW(Servo1PIN,Servo2PIN,SleepTime):
    servo1 = GPIO.PWM(Servo1PIN, 50)
    servo2 = GPIO.PWM(Servo2PIN, 50)
    servo1.start(10.5)
    servo2.start(10.5)
    time.sleep(SleepTime)
    servo1.stop()
  servo2.stop()

#jeden serwomechanizm, niezgodnie ze wskazówkami zegara
def servo_CCW(ServoPIN,SleepTime):
    servo = GPIO.PWM(ServoPIN, 50)
    servo.start(4.5)
    time.sleep(SleepTime)
    servo.stop()

#jeden serwomechanizm, niezgodnie ze wskazówkami zegara
def dual_servo_CCW(Servo1PIN,Servo2PIN,SleepTime):
    servo1 = GPIO.PWM(Servo1PIN, 50)
    servo2 = GPIO.PWM(Servo2PIN, 50)
    servo1.start(4.5)
    servo2.start(4.5)
    time.sleep(SleepTime)
    servo1.stop()
    servo2.stop()
```

Listing 2. Fragment funkcji realizujący procedurę wydania pokarmu do dozownika z lewej strony

```
if(HopperSide == „Left”):
    GPIO.output(GPIO_ButtonL_LED_PIN, False)
    GPIO.output(BeeperPin, True)
    time.sleep(.10)
    GPIO.output(BeeperPin, False)
    GPIO.output(GPIO_ButtonL_LED_PIN, True)
    print „Ok, food is coming out the left bin!”
    servo_CW(Servo1Pin,FeedTime)
```

Listing 3. Główna pętla programu

```
while True:
    now = datetime.now().strftime(„%Y-%m-%d %H:%M:%S”)
    if now == dusk:
        feedtime=datetime.now(timezone(‚US/Central’))
        print „Cat Feeding time %s” % feedtime.strftime(„%Y-%m-%d %H:%M:%S”)
        feed_cat(„Zelda”)
        feed_cat(„Thor”)
        send_email(„both”,feedtime)
        time.sleep(60)
    if time.strftime(„%H”) == „03” and time.strftime(„%M”) == „01” and time.strftime(„%S”) == „01”:
        dusk = whenisdusk()
    time.sleep(1)
```

Fotografia 2. Gotowy, zautomatyzowany, podwójny dozownik

programu. Jeśli oba serwomechanizmy mają być poruszane jednocześnie, opóźnienie to musi być wywołane w momencie, gdy oba sygnały PWM zaczęły być generowane. Kod tego fragmentu programu został umieszczony na **listingu 1**.

Ponieważ procedura karmienia obejmuje nie tylko samo uruchomienie serwomechanizmów, ale też włączenie buzzera i zapalenie diod, autor przygotował funkcję realizującą tę operację. Fragment odpowiadający za wydanie pokarmu tylko po lewej stronie został pokazany na **listingu 2**.

W końcu autor przygotował funkcje do karmienia konkretnego zwierzęcia, czy to po ręcznym naciśnięciu przycisku, czy ze względu na nadejście czasu karmienia. Warto też zwrócić uwagę na polecenie pobierające informację o zmierzchu dla danej lokalizacji, którego użyto w funkcji **whenisdusk()**:

```
def whenisdusk():
    dusk = location.
dusk(local=True, date=None)
    dusk2 = dusk.strftime(„%Y-%m-
%d %H:%M:%S”)
    return dusk2
```

Istotne jest też, że obsługa przycisków została zrealizowana poprzez wykrywanie zdarzeń na wejściach za pomocą funkcji **GPIO.add_event_detect()**:

```
GPIO.add_event_detect(GPIO_
ButtonL_PIN, GPIO.RISING,
callback=LeftFeedButton,
bouncetime=500)
GPIO.add_event_detect(GPIO_
ButtonR_PIN, GPIO.RISING,
callback=RightFeedButton,
bouncetime=500)
```

Jej pierwszy parametr określa monitorowane wyprowadzenie, drugi rodzaj wykrywanego zdarzenia (zmiana sygnału z 0 na 1), trzeci precyzuje funkcję, która ma być wywołana w przypadku wystąpienia zdarzenia, a czwarty określa czas opóźnienia po wystąpieniu zdarzenia, po którym to zmiana sygnału na wejściu będzie mogła ponownie uruchomić procedury obsługi.

W głównej pętli programu pobierany jest aktualny czas i sprawdzane jest, czy nie jest to akurat chwila zmierzchu. Jeśli tak, wydawane jest polecenie nakarmienia zwierząt oraz wysłania emaila z informacją o wykonanej operacji.

Ponadto program w pętli głównej sprawdza, czy jest akurat godzina 3:01 i jeśli jest to prawda, to aktualizuje informację o godzinie zmierzchu dla danego dnia. Pętla jest wykonywana w nieskończoność, przy czym każde kolejne wykonanie opóźniane jest o jedną sekundę. Kod głównej pętli pokazano na **listingu 3**.

Podsumowanie i ocenia projektu

Zaprezentowany projekt jest dobrym przykładem tego, jak za pomocą Raspberry Pi można zautomatyzować pewne operacje, wykonywane z użyciem serwomechanizmów. Prosta instalacja oraz niezbyt skomplikowany kod programu nie wymagają dużych nakładów pracy, by uzyskać całkiem zadowalający efekt. Program można by było rozbudować o funkcję zdalnego sterowania, gdyż w obecnej wersji interfejs sieciowy jest wykorzystywany tylko i wyłączenie do sprawdzania czasu zmierzchu i wysyłania emaili z powiadomieniami (oraz do automatycznej aktualizacji czasu w trakcie pracy i do pobierania pakietów oprogramowania po instalacji systemu). Autor dobrze wykorzystał zdarzenia do monitorowania stanu wejść, do których podłączono przyciski, ale cały projekt jest nieodporny na problemy związane z brakiem determinizmu czasowego w Linuksie uruchamianym na Raspberry Pi.

Główną wadą programu jest prosty mechanizm sprawdzania, czy nadszedł odpowiedni moment karmienia. Urządzenie porównuje aktualny czas z zadanym momentem karmienia z dokładnością co do sekundy, a opóźnienie głównej pętli również trwa sekundę. Biorąc pod uwagę dodatkowo fakt, że interpreter Pythona i sam system operacyjny mogą w zupełnie niespodziewanej chwili zacząć wykonywać jakieś dodatkowe operacje (np. czyszczenie nieużywanej pamięci czy sprawdzanie jakichś aktualizacji), istnieje ryzyko, że danego pętla główna przeskoczy chwilę karmienia i uruchomi się np. sekundę przed i sekundę po zadanym momencie. Sprawi to, że tego dnia komputer nie wykryje, że nastąpił czas karmienia i je pominie, pozostawiając koty głodne. O ile właściciel nie zorientuje się samodzielnie, że nie dostał informacji o kamieniu, może to stanowić duży problem. Dlatego wypadałoby rozbudować program o jakiś mechanizm zabezpieczenia, sprawdzający czy zwierzęta zostały nakarmione danego wieczora, albo przynajmniej zmodyfikować sposób sprawdzania czy nastąpił właśnie zmierzch. Wszak wystarczy porównywać czas z dokładnością do minuty i skorzystać z 60-sekundowego opóźnienia, które i tak autor wprowadził na koniec fazy automatycznego karmienia, by przypadkiem nie wydać dwóch porcji pokarmu pod rząd.

Marcin Karbowniczek, EP

Committed to excellence

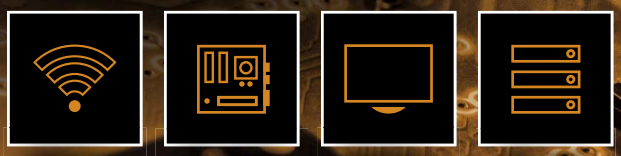
RUTRONIK SMART

Sensors | Wireless | Microcontrollers | Power Management

RUTRONIK SMART & RUTRONIK EMBEDDED

Internet of Things Seminars

RUTRONIK EMBEDDED



Wireless | Boards & Systems | Displays | Storage

Oferta **RUTRONIK SMART** skierowana jest szczególne dla producentów urządzeń z naciskiem na mniejsze i często przenośne urządzenia znajdujace zastosowanie w różnych dziedzinach. Produkty z oferty działu **RUTRONIK EMBEDDED** są zoptymalizowane pod kątem profesjonalnych wymagań, takich jak niezawodność, trwałość oraz oferują wysoki poziom integracji.

Odwiedź Seminarium Rutronik Internet of Things i dowiedz się więcej o innowacyjnych produktach i technologiach oferowanych przez trzydziestu wiodących dostawców z naszego portfolio. Odkryj szeroką gamę wyswietlaczy, modułów bezprzewodowych, systemów przechowywania danych i komponentów do zastosowań wbudowanych, oraz czujników wszystkich technologii z oferty Rutronik. Zapoznaj się również z elementami do zarządzania energią i mikrokontrolerami dla aplikacji inteligentnych.

Zapraszamy do udziału w naszym bezpłatnym Seminarium IoT **(Warszawa, 13.10.2015).**

Rejestracja:
www.rutronik.com/IoT_Seminars_2015

Register now!

www.rutronik.com

RUTRONIK
ELECTRONICS WORLDWIDE