|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Uniwersytet Morski w Gdyni** | | |  | Nr ewidencyjny ..................................... | | | |  |
| **Wydział Elektryczny** | | |  | Data złożenia pracy .............................. | | | |  |
| **Katedra Telekomunikacji Morskiej** | | |  | Nr albumu …………............................... | | | |  |
| **PRACA DYPLOMOWA INŻYNIERSKA** | | | | | | | |  |
|  | | | | | | | |  |
| Dyplomant: | inż. **Dominika Wróblewska** | | | | | | | |
| Specjalność: | Systemy i Sieci Teleinformatyczne | | | | | | | |
| Promotor: | dr. inż. Piotr Kaczorek | | | | | | Ocena: | |
| Recenzent: |  | | | | | | Ocena: | |
| Egzamin dyplomowy: | | | Data: | | | Ocena: | |
| Temat: | | | | | | | |  |
| **Wykorzystanie języka Python i platformy Django do tworzenia systemów pomiarowo-kontrolnych na bazie Raspberry Pi** | | | | | | | |  |
| Recenzent: .................................. | | Promotor: ..................................... | | | Dziekan:........................................ | | |  |
| Dyplomant: .................................. | |  | | |  | | |  |
| **Gdynia 2021** | | | | | | | |  |

Dominika Wróblewska Gdynia, dnia ……………...…………r.

*Imię i nazwisko*

**OŚWIADCZENIE**

Świadomy/a odpowiedzialności prawnej oświadczam, że złożona praca inżynierska pt.:

Wykorzystanie języka Python i platformy Django do tworzenia systemów pomiarowo-kontrolnych na bazie Raspberry Pi została napisana przeze mnie samodzielnie.

Równocześnie oświadczam, że w pracy wykorzystano tylko cytowaną literaturę a więc praca nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. 1994, nr 24, poz. 83) oraz dóbr osobistych chronionych prawem cywilnym.

Ponadto praca nie zawiera informacji i danych uzyskanych w sposób nielegalny i nie była wcześniej przedmiotem innych procedur urzędowych związanych z uzyskaniem dyplomów lub tytułów zawodowych uczelni wyższej.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną   
na płycie CD.

Na podstawie art. 75 §2 kodeksu postępowania administracyjnego wnoszę o odebranie tego oświadczenia jako dowodu prawdziwości okoliczności w nim podatnych, przy czym jestem świadomy odpowiedzialności karnej   
z art. 233 §1 i §6 k.k. za złożenie fałszywego oświadczenia.

………………………………………………… *podpis*

**Spis treści**

[Wstęp 4](#_Toc114843690)

1. Technologie wykorzystane w pracy 5

[1.1](#_Toc114843693) Język Python 5,6

1.2 Platforma Django 6

[1.3](#_Toc114843694) Baza danych PostgreSQL 0

[1.4](#_Toc114843694) Cron 0

2. Raspberry pi, wykorzystane czujniki i elementy elektroniczne 0

[2.1](#_Toc114843693) Raspberry pi 0

2.2 Czujnik DHT11 temperatury i wilgotności powietrza 0

[2.3](#_Toc114843694) Moduł pomiaru wilgotności gleby LM393 z potencjometrem 0

[2.4](#_Toc114843694) Czujnik płomieni 760-1100nm 0

[2.5](#_Toc114843694) Czujnik ruchu PIR HC-SR501 0

[2.6](#_Toc114843694) Moduł radiowy nadajnik FS100A + odbiornik 433 MHz 0

2.7 Moduł przekaźnika z 4 kanałami 0

3. Projekt 0

[3.1](#_Toc114843693) [Założenia](#_Toc114843691) projektowe 0

[3.2](#_Toc114843693) Realizacja projektu 0

4. [Funkcjonalność](#_Toc114843691) systemu z punktu widzenia użytkownika 0

5. Podsumowanie 0

Bibliografia 0

[Wykaz](#_Toc114843691) rysunków 0

[Wykaz](#_Toc114843691) listingów 0

**Wstęp**

Świat dąży do coraz większej automatyzacji. Próbuje się automatyzować pracę, samochody ale i także codzienne czynności. Wszystko dzięki zastosowaniu różnych aplikacji informatycznych, które pomagają w codziennym życiu. Systemy takie pozwalają analizować, przetwarzać dane oraz dają możliwość zdalnego sterowania różnymi urządzaniami elektronicznymi. Automatyczne zbieranie umożliwia łatwiejszą analizę oraz podgląd danych w czasie teraźniejszym np. czujników, które znajdują się w oddali od użytkownika.

Aplikację internetowe mają bardzo dużo zastosowań, automatyzacja oraz zbieranie danych są jednymi z nich. Ważną cechą aplikacji jest interaktywność. Klasyczna strona internetowa ma charakter bardziej informacyjny. Dzięki aplikacji użytkownik jest w stanie sterować elementami elektronicznymi bezprzewodowo.

Celem pracy było wykonanie systemu pomiarowo kontrolnego, zbierającego informacje pomiarowe z czujników oraz dającego możliwość sterowania elementami elektronicznymi podłączonymi do mikroprocesora Rasberry Pi za pomocą aplikacji internetowej napisanej w języku Python używając platformy Django. Stworzona przez autora aplikacja to system działającym na zasadzie „Smart Home”, tzn. użytkownik ma możliwość kontroli oraz monitorowania swojego domu, nawet będąc poza nim.

Pierwsza część pracy to opis zawierający wykorzystane technologie, narzędzia oraz wzorce użyte podczas projektu. Natomiast w drugiej części znajdują się opisane wszystkie elementy elektroniczne oraz czujniki opisane na podstawie dokumentacji. W trzeciej części przedstawione zostaną założenia postawione przed przystąpieniem do projektu. W czwartej części zostanie przedstawiony zaprojektowany obwód elektryczny, opiane struktura aplikacji wraz z opisem folderów i plików tworzących system oraz kod przedstawiający wybrane elementy. W piątej części zaprezentowany zostanie wygląd stworzonego systemu „Smart Home” od strony użytkownika składający się z zrzutów ekranu przedstawiających wszystkie funkcje oraz ich opis. Ostatnia część będzie podsumowaniem wykonanej pracy.

1. **Technologie wykorzystane w pracy**

Do powstania systemu niezbędne są trzy składniki: aplikacja, baza danych oraz mikroprocesor połączony z odpowiednimi czujnikami. Aplikacje webowe są pisane w wielu językach jak np. Python, PHP, Ruby lub Java. Ze względu na temat pracy oraz lepszą znajomość autora system został stworzony w języku Python wykorzystując najbardziej popularną platformę webową jaką jest Django. Podczas tworzenia aplikacji najnowszą wersją języka programowania był Python 3.8.6, a platformy Django 3.1.3 i taka wersja została zastosowana. Aby móc zapisywać aktualny stan czujników potrzebna była baza danych. Autor stworzył bazę danych PostgreSQL, którą połączył w ustawieniach używanej platformy, dzięki czemu tabele są pobierane od razu do modeli w aplikacji.

Do stworzenia systemu potrzebne jest odpowiednie środowisko programistyczne. Autor wybrał popularne oprogramowanie jakim jest Visual Studio Code, który umożliwia nie tylko pisanie aplikacji w języku Python, ale również w językach HTML, CSS oraz JavaScripct.

Proces tworzenia stron internetowych jest dzielony na dwie części:

- Frontend jest to cześć, którą widzi użytkownik po wpisaniu adresu aplikacji. Jest pisany w języku HTML, stylizowany arkuszem stylów CSS oraz dzięki zastosowaniu języka JavaSript jest nadawana dynamika strony.

- Backend to cześć, która odpowiada za zarządzanie są tam pobierane dane z bazy danych, obrabiane do potrzebnych formatów i przekierowywane fronednu, dzięki odpowiedniemu językowi szablonów. W platformie Django jest to język szablonów Django, jednak jest on bardzo podobny do szablonów takich jak jinja2 czy Smarty.

**1.1 Język Python**

Python to szeroko stosowany obiektowy język programowania wysokiego poziomu z dynamiczną semantyką, który służy do programowania ogólnego. Został stworzony w 1991 roku przez Guido van Rossum. Python zawiera system typów i automatyczne zarządzanie pamięcią, obsługuje wiele paradygmatów programowania, tzn. obiektowe, imperatywne, funkcjonalne oraz style proceduralne. Język ten posiada bardzo dużo bibliotek. Składnia Pythona jest łatwa do nauczenia oraz bardzo czytelna, więc edycja nawet nie swojego kodu nie sprawia trudności jak w przypadku niektórych języków. Interpreter również ułatwia naukę, dzięki interaktywnym podpowiedziom. Patrząc na kod napisany w Pythonie od razu można odgadnąć jaki to język, ponieważ wyróżnia się strukturą m.in. nieużywane są okrągłe nawiasy i tzw. klamry przy stosowaniu pętli i warunków oraz wartości logiczne typu „boolean” są zapisywane z dużych liter.

Python jest nazywany językiem dynamicznym, ponieważ:

* sam rozróżnia typy zmiennych,
* funkcje oraz metody odbywają się w czasie wykonywania,
* posiada interaktywny interpreter

W języku Python spotykane są dwie wersje:

* Python 3.x to aktualna wersja, która jest ciągle rozwijana.
* Python 2.x to starsza wersja i żadne nowe funkcje nie są wdrażane od 2020 roku.

W wersji 2.x znaki są ograniczone do kodowania ASCII, natomiast w domyślnie pliki źródłowe Pythona w wersji 3.x są zakodowane w UTF-8. Zastosowanie tego szyfrowania w nowej wersji daje możliwość używania większości języków na świecie w zmiennych tekstowych oraz komentarzach. Kod jest automatycznie kompilowany do kodu bajtowego i wykonywany.

Python jest bardzo szybkim językiem, ponieważ nie ma etapu kompilacji. Interpreter bardzo dokładnie zgłasza błędy w kodzie, co daje możliwość sprawnych poprawek kodu. Dzięki dużej popularności języka, większość problemów z jakim spotykają się programiści jest rozwiązane na takich portalach jak Stack Overflow.

Kładąc duży nacisk na szybkość i czytelność kodu coraz częściej Python jest wybieranym językiem do tworzenia aplikacji internetowych. Istnieje parę platform internetowych, dzięki których łatwo buduje się serwisy webowe, np. Flask, Django czy Web2py.

**1.2 Platforma Django**

Na samym początku wszystkie strony internetowe były tworzone ręcznie, co powodowało powstawanie licznych błędów i wymagało poświęcenia bardzo dużo czasu. W 2000 roku zostały wprowadzone narzędzia sieciowe, które pomagają w budowaniu stron internetowych, narzędzia te nazwano frameworkami. Jednym z nich jest właśnie Django.

Django to platforma sieciowa napisana w języku Python, która umożliwia czyste i szybkie tworzenia aplikacji internetowych. Została stworzona w 2003 roku przez programistów sieci Web: Adriana Holovaty i Simona Willisona. Platforma ta powstała z myślą aby tworzyć szybko aplikacje, nie marnując czasu na powtarzający się kod. Oficjalnie Framework wyszedł w 2005 roku jako platforma „open source” a jego nazwa wzięła się na cześć gitarzysty Django Reinhardt.

Platforma Django ma wiele już wbudowanych funkcji, które umożliwiają:

* Routing adresów URL
* Uwierzytelnianie
* Migracje schematów bazy danych
* Mapowanie relacyjno-obiektowe

Aby zacząć projekt w tej platformie potrzebny jest tylko dowolny edytor kodu oraz terminal. Autor użył środowiska Visual Studio Code. Do uruchomienia projekt u należy skorzystać z narzędzia djano-admin służącego do zadań administracyjnych, jest instalowany wraz z biblioteką django. Polecenie „django-admin startproject mysite” użyte w terminalu automatycznie generuje projekt o nazwie mysite oraz potrzebne pliki i foldery. Wygnerowane pliki to:

* \_\_init\_\_.py: oznacza, że katalog, w którym znajduje się ten plik to katalog pakietów Python.
* settings.py: oznacza plik, w którym ustawia się cała konfiguracje aplikacji.
* urls.py: w tym miejscu są przypisywane wszystkie adresy stworzonej aplikacji oraz nazwy przypisanych do nich funkcji.
* wsgi.py: plik wsgi, który pomaga uruchamiać aplikacje na serwerze produkcyjnym.

Za pomocą pliku konfiguracyjnego Django daje możliwość połączenia się z dowolną bazą danych. Autor użyl bazy danych PostgreSQL, która została opisana w następnym rozdziale.

**1.3 Baza danych PostgreSQL**

PostgreSQL to obiektowo-relacyjny system zarządzania bazami danych (ORDBMS) oparty na POSTGRES, opracowany na University of California na Berkeley Computer Science Department. POSTGRES był pionierem wielu koncepcji, które stały się dostępne tylko w niektórych komercyjnych bazach danych znacznie później. System zarządzania obiektowo-relacyjną bazą danych, obecnie znany jako PostgreSQL, wywodzi się z pakietu POSTGRES napisanego na Uniwersytecie Kalifornijskim w Berkeley. Mając za sobą ponad dwie dekady rozwoju, PostgreSQL jest obecnie najbardziej zaawansowaną dostępną wszędzie bazą danych typu open source.

**1.4 Cron**

Cron to demon, który działa nieprzerwanie w tle na komputerze z systemem Linux. Jego zadaniem jest automatyczne uruchamianie zadań zgodnie z harmonogramem, który również znajduje się na komputerze. Harmonogram jest zawarty w pliku tekstowym ASCII o nazwie crontab („tabela cron”). Każdy użytkownik komputera ma własną tabelę crontab. Istnieją trzy elementy umożliwiające uruchomienie crona: cron musi być zainstalowany, cron musi być uruchomiony, a sam plik crontab musi istnieć. Cron to usługa, która uruchamia się automatycznie podczas uruchamiania. Jak wspomniano, crontab to po prostu plik tekstowy z zadaniami wymienionymi w oddzielnych wierszach.

Cronjob to wiersz zawierający sześć pól oddzielonych spacjami. Wartości w pierwszych pięciu polach określają, kiedy zadanie zostanie uruchomiony, a szóste pole definiuje samo zadanie. Pierwsze pięć pól dotyczy minut, godzin, dni, miesięcy i dni powszednie. Przykładowe polecenia w crontabie:

* „\* \* \* \* \* python3 task.py” – oznacza, że co minutę będzie uruchamiany za pomocą języka python skrypt task.py
* „0 \* \* \* \* python3 task.py” – oznacza, że o każdej równej godzinie uruchamiany za pomocą języka python skrypt task.py
* „30 16 \* \* 1-5 python3 task.py” – oznacza, że od poniedziałku do piątku o 16:30 będzie uruchamiany za pomocą języka python skrypt task.py
* „0 10 1 12 \* python3 task.py” – oznacza, co roku 1 grudnia o 10.00 bedzię uruchamiany za pomocą języka python skrypt task.py

1. **Raspberry pi, wykorzystane czujniki i elementy elektroniczne**

Poza oprogramowaniem również potrzebny jest mikrokontroler obsługujący czujniki i różne elementy elektroniczne, które będą wykonywać swoje określone funkcje.

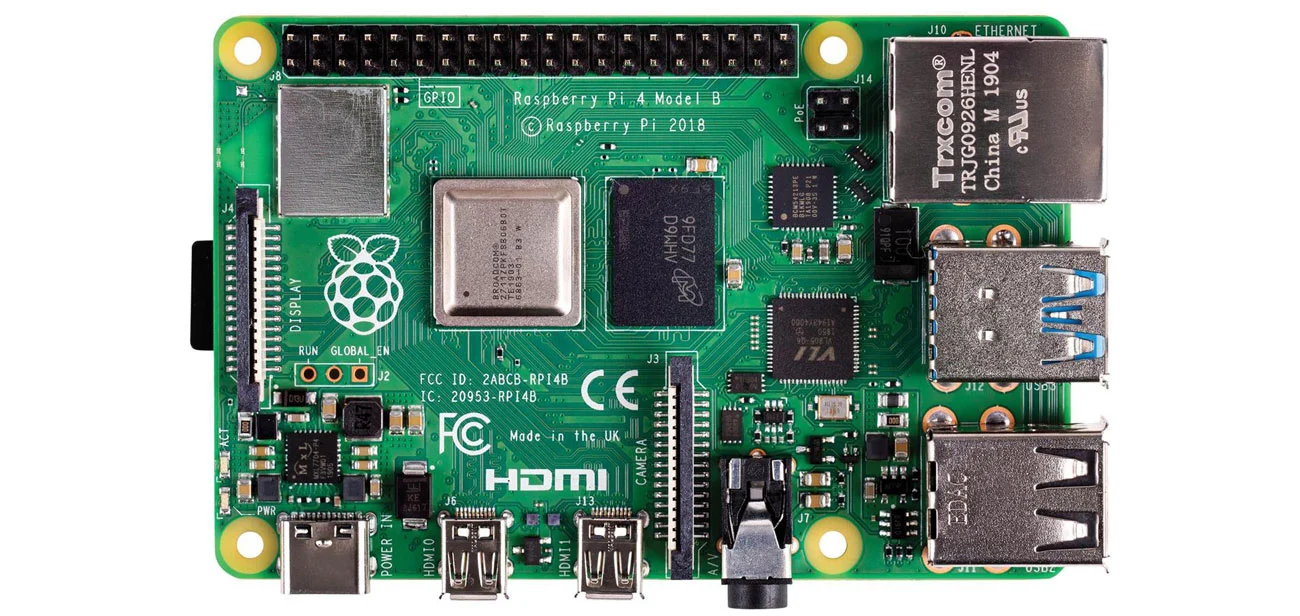
**2.1 Raspberry Pi**

Raspberry Pi to komputer jednopłytkowy wielkości karty kredytowej opracowany przez Raspberry Pi Foundation, Wielka Brytania. Płytka jest miniaturowym cudem, ma ekstremalną moc obliczeniową i jest w stanie tworzyć niesamowite projekty. Plik komputer kosztuje od 5 do 35 USD i jest idealny do wykonywania wszelkiego rodzaju zadań obliczeniowych i łączenia różnych rodzajów urządzeń przez GPIO.

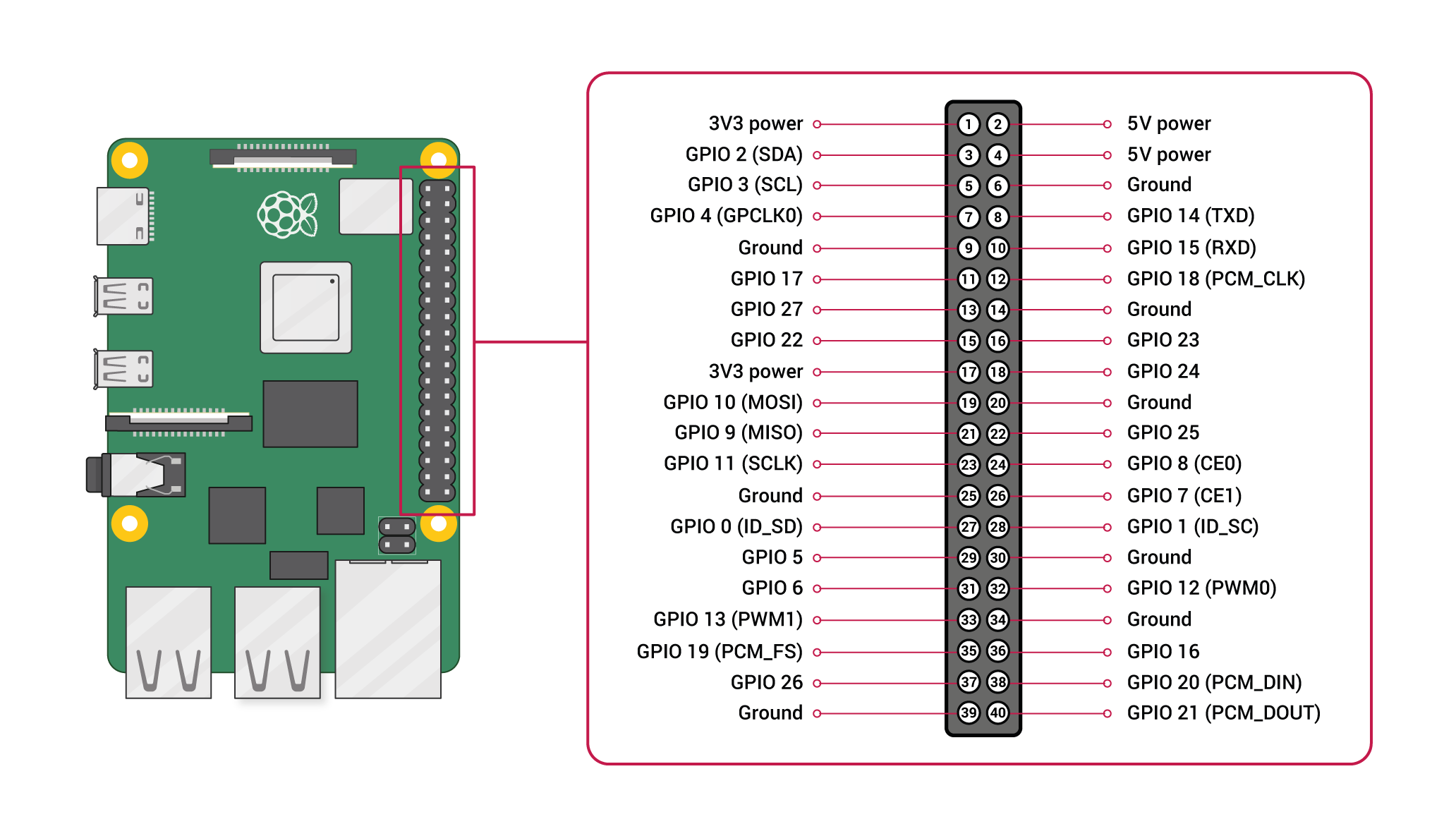
Płyta Raspberry Pi zawiera procesor ARM oparty na Broadcom, układ graficzny, pamięć RAM, GPIO i inne złącza dla urządzenia zewnętrzne. Procedura obsługi Raspberry Pi jest bardzo podobna w porównaniu z komputerem PC i wymaga dodatkowych sprzęt taki jak klawiatura, mysz, wyświetlacz, zasilacz, karta SD z zainstalowanym systemem operacyjnym (działającym jak dysk twardy) do działania. Raspberry Pi obsługuje również porty USB, Ethernet dla połączeń Internet / Network-Peer to Peer.

Jak każdy inny komputer, na którym system operacyjny działa jako podstawa działania. Raspberry Pi, ułatwia open source system operacyjny oparty na Linuksie. Do tej pory zostało uruchomionych ponad 30 systemów operacyjnych opartych na różnych odmianach Linuksa. Fundacja Raspberry Pi wypuściła również różne akcesoria, takie jak Camera, Gertboard i Compute Model Kit do wdrażania dodatkowych modułów sprzętowych.

W projekcie został użyty model Raspberry Pi 4B, który został przedstawiony na rys. 1.



Rys. 1 Zrzut z góry na płytkę Raspberry Pi 4B



Rys. 2 Raspberry Pi 4B opis pinów

**2.2 Czujnik DHT11 temperatury i wilgotności powietrza**

Czujnik temperatury i wilgotności DHT11 zawiera kompleks czujników temperatury i wilgotności z kalibrowanym cyfrowym wyjściem sygnału. Korzystając z ekskluzywnej akwizycji sygnału cyfrowego technika oraz technologia wykrywania temperatury i wilgotności, zapewnia wysoką niezawodność i doskonałą długoterminową stabilność. Ten czujnik zawiera rezystancyjny element do pomiaru wilgotności oraz element do pomiaru temperatury NTC i łączy się z wysokowydajnym 8-bitowym mikrokontrolerem, oferując doskonałą jakość, szybką odpowiedź, zdolność przeciwdziałania zakłóceniom i opłacalność.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rys. 3 Czujnik DHT11

Do obsługi czujnika można użyć takiej biblioteki jak dht11.

**2.3 Moduł pomiaru wilgotności gleby LM393 z potencjometrem**

Czujnik wilgotności gleby mierzy zawartość wody w glebie. Moduł wykorzystuje komparator LM393 do porównania poziomu wilgotności gleby z zadanym progiem. Kiedy moduł deficytu wilgotności gleby generuje wysoki poziom i odwrotnie. Ponieważ woda jest ważnym zasobem i nie wszystkie społeczności na całym świecie mogą sobie pozwolić na liberalne podejście do swoich potrzeb wodnych; Stało się ważne, aby jak najbardziej efektywnie wykorzystywać dostępną wodę, zwłaszcza w rolnictwie. W celu ograniczenia nadmiernego nawadniania upraw można zastosować nienadzorowany czujnik wilgotności gruntu, który mierzy aktualny poziom wilgotności w gleba otaczająca rośliny. Dzięki temu rolnik będzie wiedział, kiedy należy podlewać / przestać podlewać swoją uprawę. Dla wygody informacje o wilgotności powinny być przesyłane bezprzewodowo do użytkownika. Omówiono projekt nienadzorowanego czujnika wilgotności gruntu oraz systemu komunikacji bezprzewodowej / interfejsu użytkownika. Konstrukcja czujnika składa się z mostka Wheatstone'a do określania rezystancji gruntu, a następnie ze wzmacniacza różnicowego do zamiany zmierzonego oporu na napięcie. Dzieje się tak, ponieważ istnieje korelacja między wilgocią a oporem. Napięcie to jest interpretowane przez mikrokontroler jako dane dotyczące wilgotności i przesyłane bezprzewodowo do węzła odbiorczego komunikacji Lora przez węzeł nadawczy komunikacji Lora. Następnie węzeł odbiorczy przekazuje te informacje do hosta komputera PC w celu uzyskania dostępu przez użytkownika. System posiada obwód mocy, który składa się z baterii i regulatora liniowego.

**2.4 Czujnik płomieni 760-1100nm**

Niewielki moduł optycznego czujnika płomieni marki Waveshare, przeznaczony do współpracy z dowolnym mikrokontrolerem, zasilanym napięciem od 3,3 V do 5 V. Działanie sensora opiera się na detekcji promieniowania podczerwonego w zakresie od 760 nm do 1100 nm, generowanego przez ogrzane masy powietrza, powstające w wyniku procesów spalania.

Prezentowany czujnik płomienia posiada dwa wyjścia – analogowe (wyprowadzone bezpośrednio w wyjścia dzielnika napięciowego, zawierającego fotodetektor) oraz cyfrowe (zapewniane przez komparator LM393 z przestrajanym za pomocą potencjometru progiem zadziałania). Zmiana stanu wyjścia jest dodatkowo sygnalizowana przez wbudowaną diodę LED.

Czujnik płomieni 760-1100nm - analogowy - Waveshare 9521 stanowi świetny przykład zastosowania fotodetektorów, pracujących w paśmie podczerwieni, z płytkami Arduino, STM32 i in. Pozwala wykrywać nie tylko sam płomień, ale także inne źródła promieniowania IR, których widmowy pik emisji mieści się w zakresie czułości spektralnej detektora.

Należy pamiętać, że czujnika nie należy stosować jako elementu systemów ostrzegania, od których reakcji zależy np. podjęcie akcji ewakuacyjnej lub wezwanie służb ratunkowych. Do takich zastosowań polecamy dostępne w naszej ofercie, specjalistyczne detektory pożaru i/lub dymu, posiadające niezbędne atesty i certyfikaty bezpieczeństwa, potwierdzone restrykcyjnymi badaniami laboratoryjnymi.

**2.4 Czujnik ruchu PIR HC-SR501**

Moduł typu PIR - Passive Infrared Sensor, co w języku polskim znaczy pasywny czujnik podczerwieni, służący do wykrywania ruchu, który jest sygnalizowany stanem wysokim. Zasadniczo czujniki PIR składają się z piroelektrycznego czujnika (który można zobaczyć poniżej jako okrągłą metalową puszkę z prostokątnym kryształem w środku), który może wykrywać poziomy promieniowania podczerwonego. Wszystko emituje promieniowanie o niskim poziomie, a im coś jest cieplejsze, tym więcej emituje promieniowania. Czujnik w czujniku ruchu jest w rzeczywistości podzielony na dwie połowy. Powodem tego jest to, że staramy się wykryć ruch (zmianę), a nie średnie poziomy IR. Dwie połówki są połączone przewodami tak, że znoszą się nawzajem. Jeśli jedna połowa widzi więcej lub mniej promieniowania podczerwonego niż druga, moc wyjściowa będzie się zmieniać wysoko lub nisko.

Czujniki PIR są bardziej skomplikowane niż wiele innych czujników opisanych w tych samouczkach (np. Fotokomórki, czujniki FSR i przełączniki pochylenia), ponieważ istnieje wiele zmiennych, które mają wpływ na wejście i wyjście czujników. Aby rozpocząć wyjaśnianie, jak działa podstawowy czujnik, posłużymy się tym dość ładnym schematem. Sam czujnik PIR ma w sobie dwa gniazda, każdy z nich jest wykonany ze specjalnego materiału wrażliwego na podczerwień. Zastosowany tutaj obiektyw tak naprawdę nie robi zbyt wiele, więc widzimy, że oba szczeliny mogą „widzieć” z pewnej odległości (zasadniczo czułość czujnika). Gdy czujnik jest w stanie spoczynku, oba gniazda wykrywają taką samą ilość podczerwieni, ilość promieniowania otoczenia wypromieniowaną z pomieszczenia, ścian lub na zewnątrz. Kiedy przechodzi ciepłe ciało, takie jak człowiek lub zwierzę, najpierw przechwytuje połowę czujnika PIR, co powoduje dodatnią różnicę między dwiema połówkami. Kiedy ciepłe ciało opuszcza obszar wykrywania, dzieje się odwrotnie, w wyniku czego czujnik generuje ujemną zmianę różnicową. Wykrywane są te impulsy zmiany.

**2.4 Moduł radiowy nadajnik FS100A + odbiornik 433 MHz**

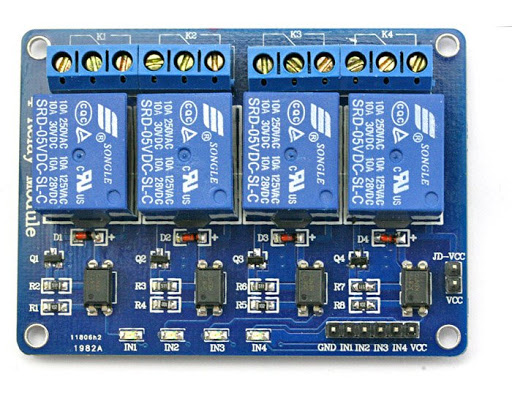
W zestawie nadajnik i odbiornik, które komunikują się ze sobą na częstotliwości 433 MHz. Urządzenia zasilane są napięciem 5 V, komunikacja z mikrokontrolerem odbywa się poprzez interfejs jednoprzewodowy.

Moduł bezprzewodowy 433 MHz jest jednym z tanich i łatwych w użyciu modułów dla wszystkich projektów bezprzewodowych. Moduły te mogą być używane tylko w parach i możliwa jest tylko komunikacja simplex. Oznacza to, że nadajnik może przesyłać tylko informacje, a odbiornik tylko je odbierać, więc można przesyłać dane tylko z punktu A do B, a nie z B do A. teoretycznie może osiągnąć nawet 100 metrów. Ale praktycznie nie możemy uzyskać około 30-35 metrów w normalnych warunkach testowych, więc jeśli szukasz prostej komunikacji bezprzewodowej do przesyłania informacji na krótką odległość, ta para RF może być właściwym wyborem. Sam moduł nie może działać samodzielnie, ponieważ wymagał pewnego rodzaju kodowania przed nadaniem i dekodowania po odebraniu; więc musi być używany z układem scalonym kodera lub dekodera lub z dowolnym mikrokontrolerem na obu końcach. Najprostszym sposobem użycia jest użycie enkodera HT12E i układu scalonego dekodera HT12D. Moduł wykorzystuje ASK (kluczowanie z przesunięciem amplitudy), dzięki czemu można go łatwo łączyć również z mikrokontrolerami.

Do obsługi nadajnika oraz odbiornika można użyc takiej biblioteki jak rpi-rf.

**2.4 Moduł przekaźnika z 4 kanałami**

Jest to 4-kanałowa płyta interfejsu przekaźnika niskiego poziomu 5 V, a każdy kanał wymaga prądu sterownika 15-20 mA. Może służyć do sterowania różnymi urządzeniami i urządzeniami o dużym natężeniu prądu. Jest wyposażony w przekaźniki wysokoprądowe pracujące pod napięciem AC250V 10A lub DC30V 10A. Posiada standardowy interfejs, którym można sterować bezpośrednio z mikrokontrolera. Ten moduł jest optycznie izolowany od strony wysokiego napięcia ze względów bezpieczeństwa, a także zapobiega pętli uziemienia podczas łączenia się z mikrokontrolerem.



1. **Projekt**

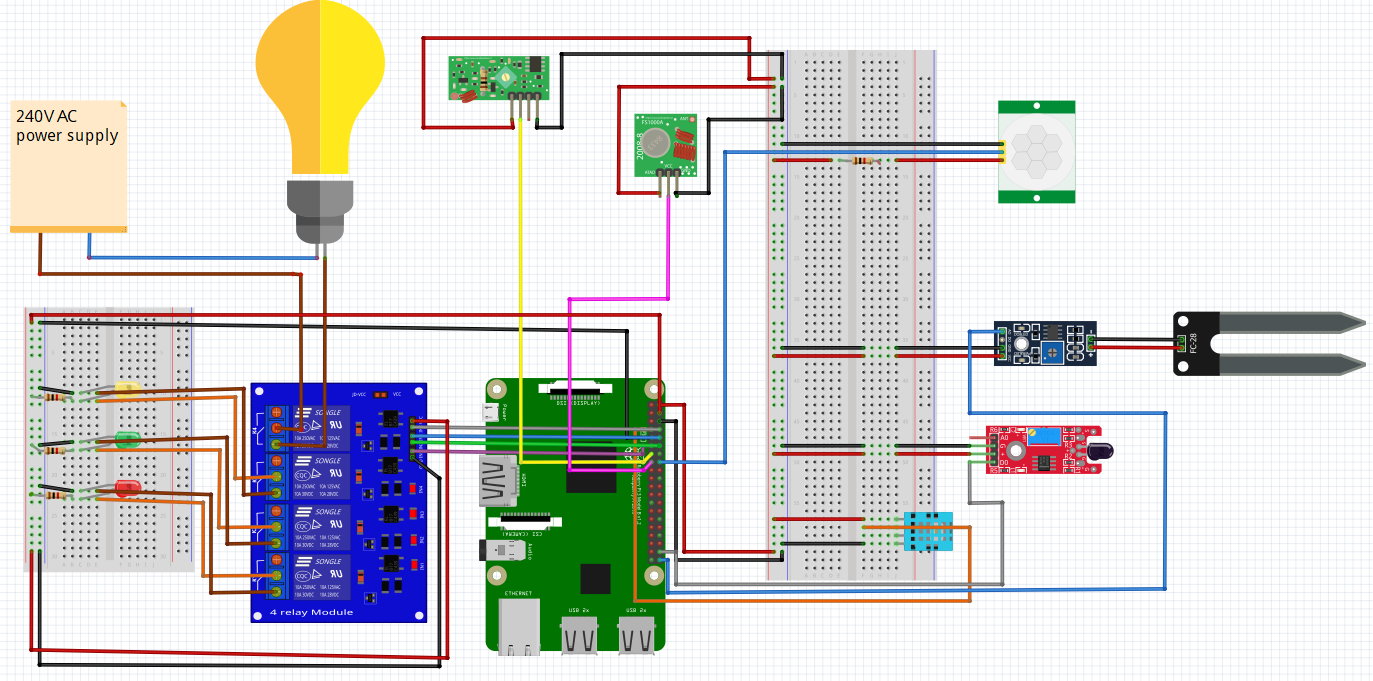
Stworzony projekt jest to system Smart Home. Umożliwia użytkownikowi zarówno obserwacje jak i kontrole swojego domu za pomocą prostego i intuicyjnego wyglądu. Projekt został stworzony do indywidualnego użytkownika, dla którego administrator stworzył jedno konto. Do aplikacji należy się zalogować, co też daje bezpieczeństwo przed wglądem do systemu przez osoby zewnętrzne.

**3.1 Założenia projektowe**

Założenia projektowym było stworzenie systemu pomiarowo-kontrolnego na bazie Raspberry Pi wykorzystując język Python i platformę Django. Autor wykonując system spełnił wszystkie cele tworząc aplikacje Smart Home.

**3.2**  **Realizacja projektu**

Pierwszym etapem podczas realizacji projektu było stworzenie schematu elektrycznego ze wszystkimi czujnikami i elementami elektronicznymi. Projekt został zaprojektowany za pomocą programu Fritzing, dla którego bardzo łatwo znaleźć do pobrania wszystkie biblioteki czujników kompatybilnych z Raspberry Pi. W schemacie została zastosowana inna wersja mikrokontrolera niż ta użyta w projekcie, z w powodu problemu autora ze znalezieniem biblioteki posiadanej płytki, jednak wykorzystywane piny obu mikrokomputerów Raspberry Pi są identyczne.



Rysunek 1 Schemat elektryczny Raspberry Pi i użytych czujników zaprojektowany w programie Fritzing

Po stworzeniu schematu można było przystąpić do pierwszego uruchomienia Raspberry Pi i zainstalowania systemu Rasbian. Następnie zostały skonfigurowane ustawienia mikro komputera oraz pobrane środowisko Visual Studio Code, najnowsza wersja języka Python i wszystkie potrzebne biblioteki do projektu, w tym platforma Django. Potem można było przystąpić do łączenia czujników oraz elementów elektronicznych na podstawie zbudowanego wcześniej schematu. Kolejno autor stworzył bazę danych PostgreSQL i następnie za pomocą komendy „django-admin startproject mysite” został utworzony projekt o nazwie „mysite”. Aby powstała aplikacja wpisano komendę „python manage.py startapp app”, która zbudowała aplikacje o nazwie „app”.

Pierwszą częścią po stworzeniu systemu należało odpowiednio skonfigurować ustawienia w pliku settings.py. Do konfiguracji została dodana nasza aplikacja, baza danych oraz dodana ścieżka plików statycznych i szablonów html. Ustawienia zostały kolejno przedstawione poniżej.

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'app'

]

Listing 1 Konfiguracja w Django zainstalowanych aplikacji

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.postgresql\_psycopg2',

        'NAME': 'postgres',

        'USER': 'USERNAME',

        'PASSWORD': 'PASSWORD',

        'HOST': '127.0.0.1',

        'PORT': '5432',

    }

}

Listing 3 Konfiguracja w Django bazy danych

STATIC\_URL = '/ static /'

TEMPLATE\_DIRS = (

    os.path.join(BASE\_DIR , 'templates'),

)

Listing 3 Konfiguracja w Django ścieżki plików statycznych i szablonów html

Po ustawieniach autor stworzył modele odpowiednie modele zapisujące stan oświetlenia, dzięki któremu aplikacja wie, który element światła jest obecnie zapalony. Model ten przedstawiono na listingu 4.

class Lights(models.Model):

    name = models.TextField()

    turn\_on = models.BooleanField()

Listing 4 Model klasy Lights określający właściwości tabeli app\_lights

Następnym elementem był model, który służy do zapisywania temperatury, wilgotności powietrza oraz daty i godziny w momencie pobrania danych. Został on przedstawiony na listingu 5.

class Temp\_And\_Hum(models.Model):

    temp = models.TextField()

    hum = models.TextField()

    date = models.DateField()

    time = models.TimeField()

Listing 5 Model klasy Temp\_And\_Hum określający właściwości tabeli app\_temp\_and\_hum

Ostatnim został stworzony model, dzięki któremu użytkownik konfiguruje czy chciałby otrzymywać powiadomienia po wykryciu ruchu, w jakiej postaci oraz aktualny numer telefonu i email, na które miałby zostać wysłane wiadomości. Został on przedstawiony na listingu 6.

class Motion\_Detector(models.Model):

    user\_fkey= models.ForeignKey(

        settings.AUTH\_USER\_MODEL,

        on\_delete=models.CASCADE)

    phone = models.TextField()

    email = models.TextField()

    accept\_sms = models.BooleanField()

    accept\_email = models.BooleanField()

    send = models.BooleanField()

Listing 6 Model klasy Motion\_Detector określający właściwości tabeli app\_motion\_detector

Po stworzeniu modeli dane zostały migrowane do bazy danych co spowodowało utworzenie tabel, dzięki czemu autor przeszedł to pisania skryptów. Zostały stworzone dwa skrypty, jeden jest odpalany za pomocą crona, który został opisany wcześniej. Jest on odpalany regularnie co godzinę, pobiera dane z czujnika dht11 i dodaje wiersz do tabeli app\_temp\_and\_hum. W skrypcie zostały napisane dwie funkcje get\_temp\_hum(), która pobiera dane z czujników dht11 oraz zwraca je oraz q\_run(query), która przyjmuje kod w języku SQL, łączy się z bazą danych i wysyła zapytanie do bazy danych. Poniżej przedstawiono kawałek kodu z tego skryptu wykorzystujący te funkcje.

temp, hum = get\_temp\_hum()

if temp != '0' and hum != '0':

    query = """INSERT INTO app\_temp\_and\_hum(temp, hum, date, time)  VALUES ('{}', '{}', now(), now())""".format(temp, hum)

    q\_run(query)

Listing 7 Najważniejszy kawałek kodu ze skryptu, który pobiera dane z czujnika dht11 i wysyła do bazy danych

Drugim skryptem był skrypt, który działa ciągle w tle. Działa on tak, że jak czujnik wykryje ruch to sprawdza w bazie danych czy jakiś użytkownik akceptuje powiadomienia sms lub email i następnie wysyła odpowiednią wiadomość i ustawia w tabeli send = True, co powoduje że trzeba zresetować przez aplikacje ten status, aby mieć możliwość ponownego otrzymania takiej informacji. Rozwiązuje to problem w przypadku wykrycia ruchu parę razy w ciągu minuty, co by powodowało masowy przypływ wiadomości. Funkcja wykrywająca ruch została przedstawiona na listingu 8.

def check\_motion\_sensor():

    while True:

        query = """SELECT accept\_sms, accept\_email, phone, email,  id FROM app\_motion\_detector WHERE send = false """

        users\_list = q\_run(query)

        i = GPIO.input(16)

        if i == 0:

            time.sleep(5)

        elif i == 1:

            for user in users\_list:

                user\_id = user[4]

                accept\_sms = user[0]

                accept\_email = user[1]

                if accept\_sms:

                    send\_sms(user[2])

                if accept\_email:

                    send\_email(user[3])

                if accept\_sms or accept\_email:

                    query = """UPDATE app\_motion\_detector set send = true WHERE id = '{}';""".format(user\_id)

                    q\_run(query)

            time.sleep(5)

Listing 8 Funkcja sprawdzająca czy czujnik wykrył ruch

Do wysyłania wiadomości sms została wykorzystany system SMSAPI, który generuje nam token, który uwierzytelnia z kontem stworzonym na tej aplikacji. Tego rodzaju relacja jest płatna. Funkcja umożliwiająca przesyłania sms przedstawiona na listingu 9.

def send\_sms(phone):

    token = "lV3Iqs8dHGTziNLacNelDwirCs5zXw6xmbJIFgNG"

    phone\_number = phone

    message = "Your sensor has detected movement. If it's not you, check the camera on the smart home app."

    client = SmsApiPlClient(access\_token=token)

    send\_results = client.sms.send(to=phone, message=message)

Listing 9 Funkcja wysyłająca sms

Do wysyłania wiadomości email zostały wykorzystane protokół SMTP oraz SSL do łączności z serwerem poczty. Do edycji wyglądu tekstu użyto biblioteka email.mime. Ta opcja jest bezpłatna, ale wymaga założenia konta na dowolnej poczcie. Autor aplikacji założył konto na portalu gmail. Funckja wysyłając emaila została przedstawiona na listingu 10.

def send\_email(email):

    sender\_email = "smarthomeinzynierka@gmail.com"

    receiver\_email = email

    password = 'inzynierka1'

    message = MIMEMultipart("alternative")

    message["Subject"] = "Smart Home: Motion Detector!"

    message["From"] = sender\_email

    message["To"] = receiver\_email

    text = """<p>Hello!<br> Your sensor has detected movement. If it's not you, check the camera on the smart home app.</p>

            <p>This e-mail was generated automatically. Please don't answer.</p>"""

    part = MIMEText(text, "html")

    message.attach(part)

    context = ssl.create\_default\_context()

    with smtplib.SMTP\_SSL('smtp.gmail.com', 465, context=context) as server:

        server.login(sender\_email, password)

        server.sendmail(sender\_email, receiver\_email, message.as\_string())

Listing 10 Funkcja wysyłająca mail

Używając automatycznego interfejsu administratora wbudowanego w django został stworzony użytkownik. Następnie autor zaczął projektować widok oraz szablon związany z logowaniem. Do autoryzacji wykorzystano funkcje wbudowaną w Django o nazwie authenticate. Model Users z użytkownikami został stworzony automatycznie. Widok logowania został przedstawiony w listingu 11.

def login(request):

    if request.method == 'POST':

        login = request.POST['login']

        password = request.POST['password']

        user = authenticate(username=login, password=password)

        if user is not None:

            auth.login(request , user)

            return redirect('/')

        else:

            messages.info(request, 'Invalid login or password')

            return redirect('/login')

    return render(request, 'login.html')

Listing 11 Funkcja widoku logowania

Po stworzeniu logowania utworzono widok wylogowania, który został przedstawiony na listingu 12.

def logout(request):

    auth.logout(request)

    return render(request, 'logout.html')

Listing 12 Funkcja widoku wylogowania

W następnym etapie zaczęły powstawać kolejne zakładki. Najpierw został stworzony jeden bazowy html, który znajduje się w każdym module strony. Widoki, które wymagają zalogowania zaczynają się od „@login\_required(login\_url='/login')”, co powoduje, że do tych funkcji nie ma dostępu użytkownik nie zalogowany. Pierwszą zakładką jest strona główna, która pobiera dane z czujników i wysyła zmienne do html, została przedstawiona na listingu 13.

@login\_required(login\_url='/login')

def home(request):

    temp, hum, water\_status, fire\_status = read\_home\_variables()

    context = {'temp' : temp, 'hum' : hum,

'water\_status' : water\_status, fire\_status' : fire\_status}

    return render(request, 'home.html', context)

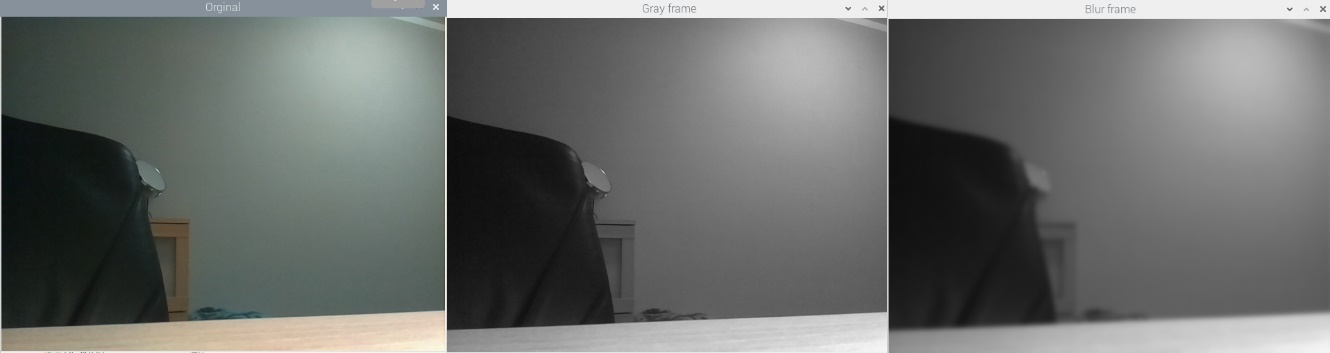
Listing 13 Funkcja widoku strony głównej

Dane na stronie głównej są pobierane co 10 sekund, jednak nie wymaga to odświeżania strony, ponieważ za pomocą języka JavaScript, dane są pobierane w tle i podmieniany jest jedynie tekst w atrybucie „th”, w którym znajdują się te wartości.

Następnie został stworzony widok, który pobiera z modelu czy obecnie zalogowany użytkownik akceptuje powiadomienia w określonej formie oraz jego numer sms i adres email i również status wysłania powiadomienia. Tak jak było wspominane wcześniej, gdy czujnik wykryje ruch wiadomość nie zostanie wysłana, jeśli już raz użytkownik taką otrzymał i nie kliknął przycisku reset. Zmiany danych jest możliwa dzięki, dzięki wykorzystaniu metody POST, która aktualizuje zasób. A wszystkie funkcje są wykonywane w bloku try, except, dzięki czemu w przypadku aktualizacji danych użytkownik dostanie informacje zwrotną czy wszystko poszło pomyślnie.

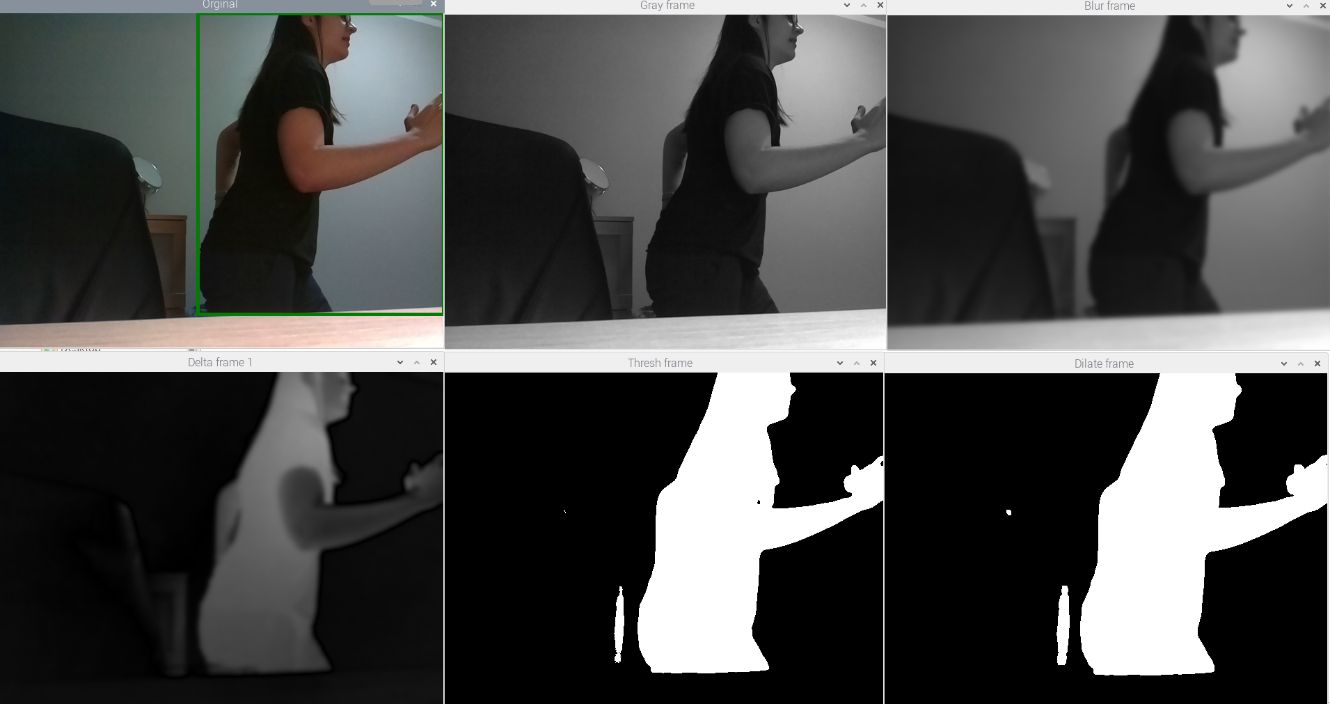
KONIEC MOTION DETECTOR – WSTAWIC LISTING

Kolejnym widokiem jest wideo na żywo. Do przechwytywania obrazu użyto biblioteki OpenCV. Obraz jest z kamery jest zbierany w postaci klatek, pierwsza po włączeniu zakładki jest zapisywana i porównywana z każdą następną. Jednak aby uzyskać dużą dokładność wykrytego ruchu ramki zostały podane kompresji. Pierwsza klatka została zamieniona na kolor szary i następnie rozmyta i zapisana do zmiennej first\_frame, co zostało pokazane na rys x.



Rysunek 2 Przedstawienie po kolei oryginalnej klatki, po przekształceniu na szaro i następna po rozmyciu

Dzięki poszarzeniu i rozmyciu obrazu kamera bez problemy odnajduje różnice pomiędzy pierwszą ramką a aktualną. Po porównaniu jest stosowany próg, który jasny obszar zamienia na wartość binarną 0, ciemny na 1, dzięki czemu uzyskujemy kontur różnic. Następnie zwiększany został biały obszar obrazu na pierwszym planie. Gdy program zaważył już widoczne różnicee został narysowany wokół zielony kwadrat. Konwersja i każdy etap porównywania obrazu został przedstawiony na rysunku x.

****

Rysunek 3 Przedstawienie krok po kroku jak obraz znajduje różnice

Obraz z narysowaną zieloną ramką, gdy wykryje ruch lub bez gdy go nie ma, zostaje zwracany w postaci bitów. Cały ten przedstawiony proces opisuje listing 15.

 def get\_frame(self):

        ret, image = self.video.read()

        gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

        gray = cv2.GaussianBlur(gray, (21,21), 0)

        if self.first\_frame is None:

            self.first\_frame = gray

        delta\_frame = cv2.absdiff(self.first\_frame, gray)

        thresh\_frame = cv2.threshold(delta\_frame, 30, 255,

cv2.THRESH\_BINARY)[1]

        thresh\_frame = cv2.dilate(thresh\_frame, None, iterations=2)

        (cnts, \_) = cv2.findContours(thresh\_frame.copy(),

cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

        for contour in cnts:

            if cv2.contourArea(contour) < 1000:

                continue

            (x, y, w, h) = cv2.boundingRect(contour)

            cv2.rectangle(image, (x, y), (x+w, y+h), (5, 122, 5), 3)

        ret, jpeg = cv2.imencode('.jpg', image)

        return jpeg.tobytes()

Listing 15 Funkcja przedstawiająca porównywanie obrazu pierwotnego do każdych następnych klatek i zwracająca obraz z namalowanym zielonym kwadratem wokół znalezionych różnić

Po kompresji obrazu, bity są zwracane do widoku transmisji video i zostają przesyłanie strumieniowo za pomocą protokołu komunikacyjnego HTTP Live Streaming. Zostało to przedstawione na listingu 16.

@gzip.gzip\_page

def camera\_streaming(request):

    try:

        return StreamingHttpResponse(gen(VideoCamera()),

content\_type="multipart/x-mixed-replace;boundary=frame")

    except:

        print("aborted")

Listing 16 Widok transmisji wideo

Widok samej kamery, do której ma dostęp użytkownik budowany jest osobno. Renderuje on jedynie szablon html.

@login\_required(login\_url='/login')

def camera(request):

    return render(request, 'camera.html')

Listing 17 Widok transmisji wideo

Udostępnienie transmisji wideo w html, jest możliwe dzięki pobieraniu ciągle nowej klatki do atrybuty <img>.

1. **Funkcjonalność systemu z punktu widzenia użytkownika**

Stworzony system ma na celu umożliwić zdalnej kontroli oraz monitorowania swojego domu za pomocą strony internetowej. Aby móc korzystać z aplikacji należy się zalogować, ponieważ gdyby do serwera została dokupiona domenę to w przypadku braku logowania każdy by miał dostęp do zarządzania domem użytkownika, dla którego ten system został napisany. Na rysunku x został przedstawiony interface logowania.

Obraz zawierający tekst, drewniane

Opis wygenerowany automatycznie

Rys. 5 Strona logowania do aplikacji

Napis Smart Home, który jest widoczny na każdej zakładce aplikacji jest indywidualnym projektem autora. Po zalogowaniu użytkownika przerzuca na stronę główną, w której u góry znajduje się 7 zakładek oraz możliwość wylogowania się. Wszystkie te przyciski są w bazowym wyglądzie i wyświetlają się po wejściu w każdą cześć projektu. To w której zakładce właśnie jesteśmy jest widoczne dzięki zakolorowaniu odpowiedniej części na kolor malinowy.

Na stronie głównej znajdują się informacje o obecnej temperaturze, wilgotności oraz czy jest wykrywany ogień i woda. Informacje są odświeżane co 10 sekund, dzięki czemu widać ochłodzenie pomieszczenia po otworzeniu okna zimą. Na rys. x przedstawiony został widok strony głównej.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rys. 6 Strona główna aplikacji

Następną zakładka to część obsługi czujnika ruchu. Użytkownik może tam ustawić jak chciałby zostać poinformowany o wykryciu ruchu w swoim domu, do wyboru jest wiadomość sms oraz mail. Poza ustawieniami rodzajem otrzymywanej informacji, również jest możliwość aktualizacji adresu email oraz numeru telefonu. Po wykryciu ruchu, gdy użytkownik dostanie wiadomość, należy wejść na stronę i zresetować status, aby móc otrzymać ponownie taką informacje. Wygląd tej części został przedstawiony na rysunku x.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rys. 7 Zakładka Motion Detector

Poniżej przedstawiono kolejno wygląd wiadomości sms oraz email informującej o wykryciu ruchu.

Obraz zawierający tekst

Opis wygenerowany automatycznie

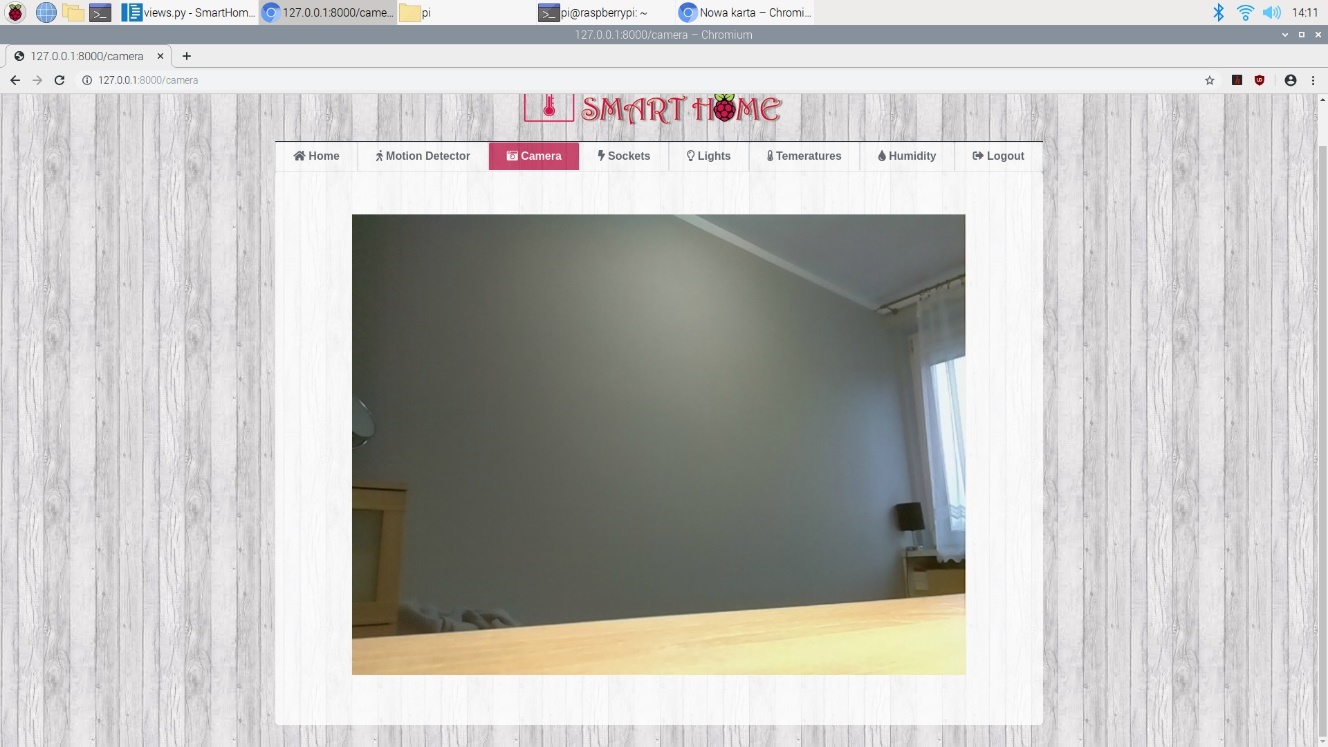
Rys. 8 Wiadomość sms po wykryciu ruchu

Obraz zawierający tekst

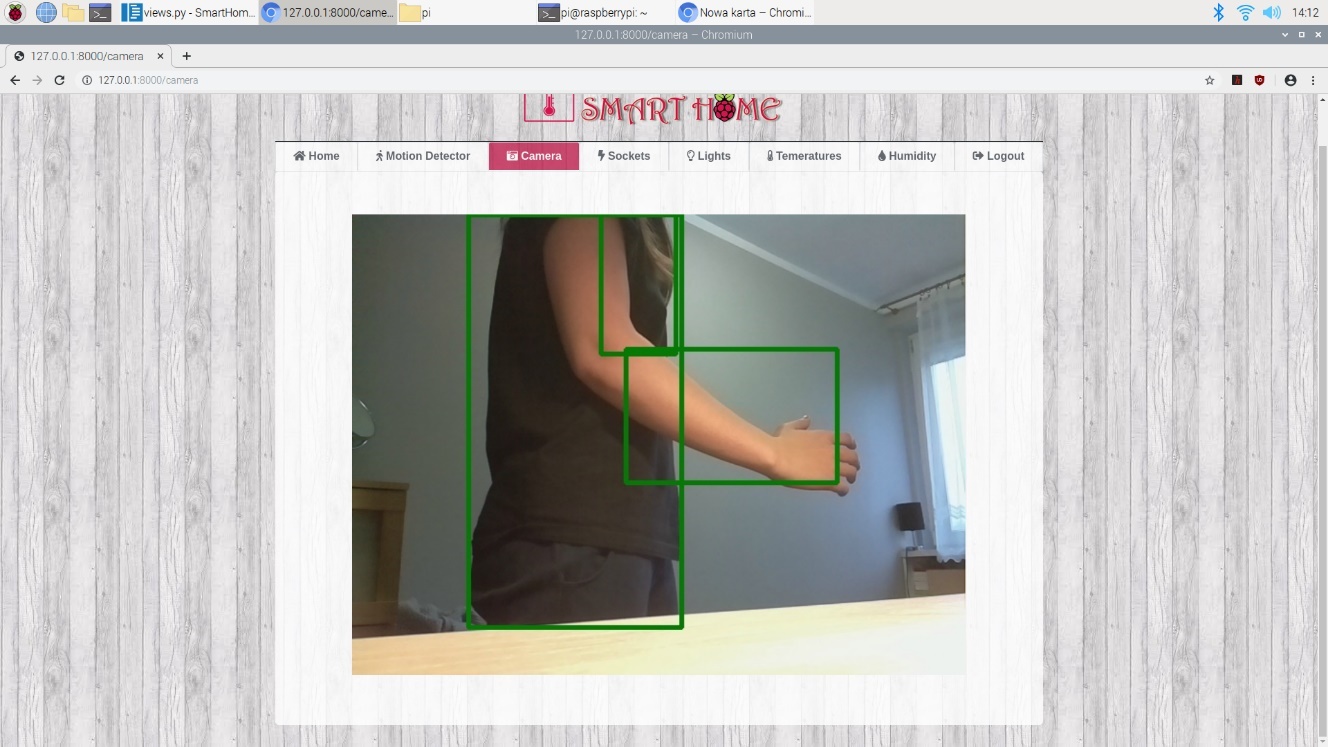
Opis wygenerowany automatycznie

Rys. 9 Widomość email po wykryciu ruchu

Następną zakładka to kamera, która umożliwia podgląd pomieszczenia, w którym się znajduje oraz wykrywa ruch. Aktywność jest zaznaczana za pomocą zielonych prostokątów. Poniżej przedstawiono kolejno kamerę z pustym pomieszczeniem oraz gdy wykryła ruch.



Rys. 10 Zakładka z kamerą w pustym pomieszczeniu



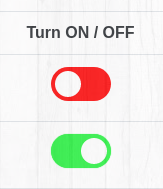
Rys. 11 Zakładka z kamerą gdy wykryto ruch

Następną zakładką jest moduł z gniazdkami bezprzewodowymi, za pomocą możemy włączać i wyłączać gniazdka. Gdy przełącznik jest na czerwono oznacza, że gniazdka są wyłączone. Został przedstawiony na rys. x.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rys. 12 Zakładka do sterowania bezprzewodowymi gniazdkami



Rys. 13 Przedstawienie wyłączonego i włączonego przełącznika

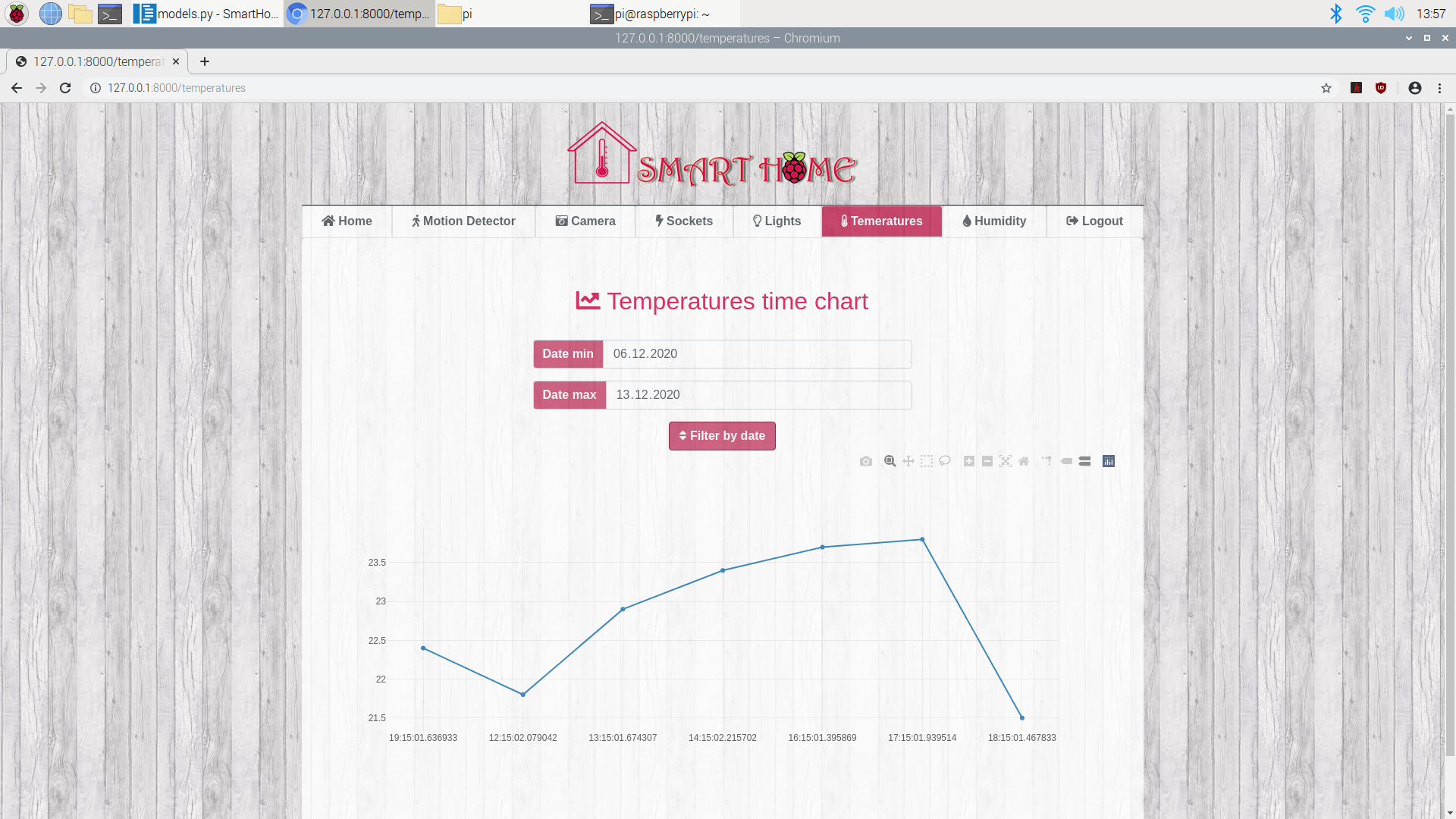
Kolejną część to sterowanie oświetleniem, jest to wizualnie bardzo podobne do zakładki z bezprzewodowymi gniazdkami, ale komunikacja aplikacja czujnik jest przewodowa co daje większą pewność wykonania w porównaniu do sterowania radiowego. Poniżej przedstawiony moduł do sterowania oświetleniem.

Obraz zawierający tekst

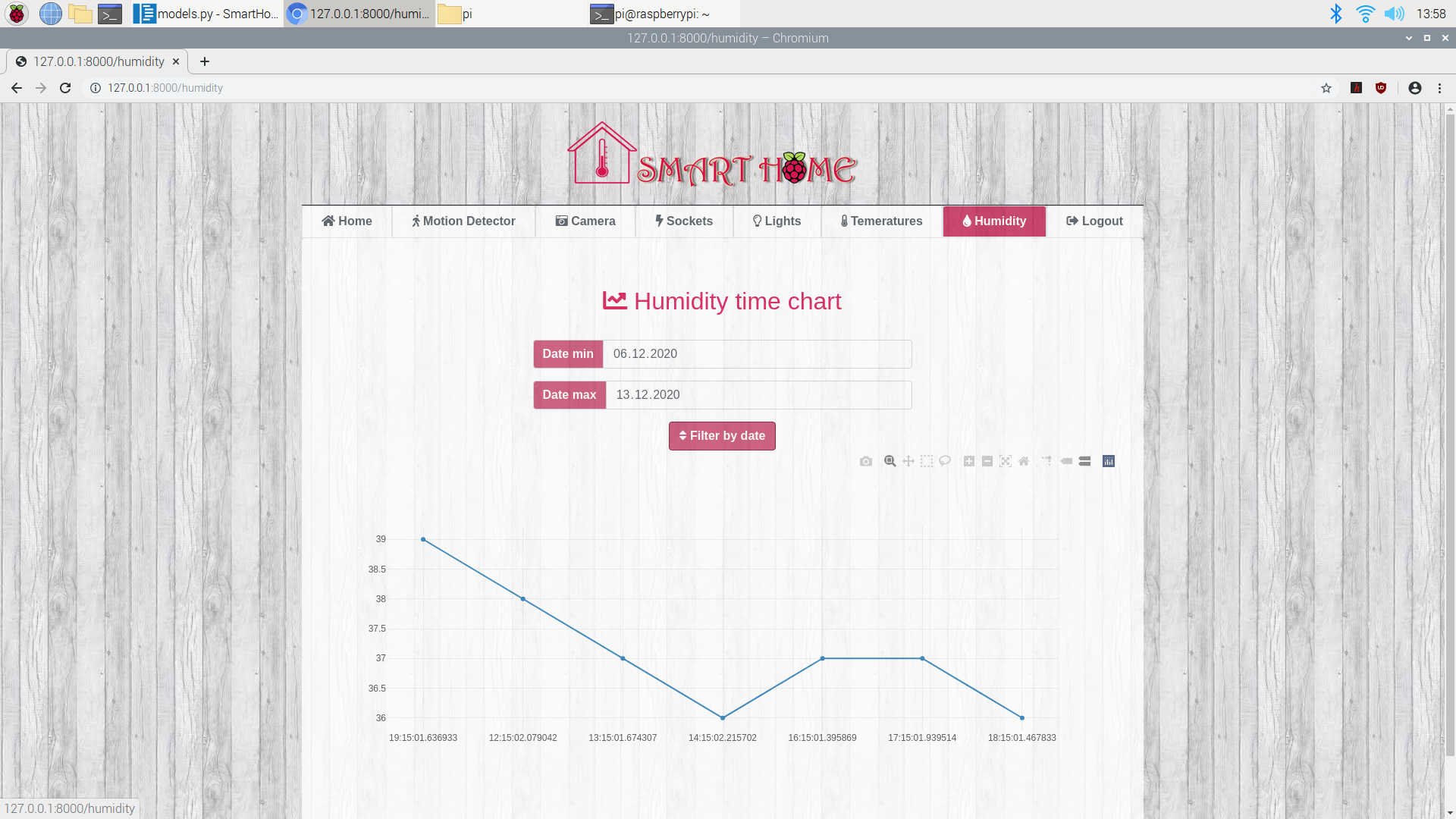
Opis wygenerowany automatycznie

Rys. 14 Zakładka do sterowania oświetleniem

Następne dwie zakładki przedstawiają wykresy temperatury oraz wilgotność powierza, dzięki którym użytkownik jest w stanie analizować jaka była temperatura w pomieszczeniu o konkretnej godzinie. Wygląd tych części został przedstawiony na rysunkach poniżej.



Rys. 15 Zakładka aplikacja przedstawiająca wykres temperatury powietrza



Rys. 16 Zakładka aplikacja przedstawiająca wykres wilgotności powietrza

Ostatni przycisk służy do wylogowania się. Po naciśnięciu przerzuca użytkownika do strony pokazanie na rys. x.

Obraz zawierający tekst, tablica suchościerna

Opis wygenerowany automatycznie

Rys. 17

Aby wrócić do aplikacji należy zalogować się ponownie.