0.1 Obsługa czujników stykowych

W systemie zostały zastosowane czujniki zbliżeniowe magnetyczne MC-38. Można je wykorzystać do określenia pozycji drzwi albo okien. Część z przewodami należy umieścić na framudze a część swobodną na drzwiach lub oknach. Obwód w części z przewodami jest domyślnie rozwarty, a po zbliżeniu magnesu zostaje zamknięty. Czujnik powinien być podłączony do wyprowadzenia GPIO i do masy, aby wykrywać zamknięcie i przerwanie obwodu. Raspberry Pi umożliwia podłączenie każdego wyprowadzenia poprzez wbudowany rezystor podciągający do napięcia zasilania (Vcc). Dzięki temu można wykrywać otwarcie okien lub drzwi jako oddalenie czujników i przerwanie obwodu – zmianę stanu odczytywaną na wyprowadzeniu z wysokiego na niski. W wykonanym prototypie dłuższy przewód podłączono do wyprowadzenia GPIO, a krótszy do masy na płytce prototypowej.

Korzystanie z wyprowadzeń GPIO w programie języku Python jest możliwe na komputerze Raspberry Pi przy użyciu pakietu RPi.GPIO. Umożliwia on odczyt stanu wyprowadzeń. Konfiguracja obsługi GPIO rozpoczyna się od określenia sposobu numeracji wyprowadzeń. Dostępne są dwie możliwości: GPIO.BOARD oraz GPIO.BCM. Pierwszy odnosi się do fizycznej lokalizacji wyprowadzeń na płytce drukowanej. Drugi sposób oznacza numeracją kanałów system-on-chip (SOC) firmy Broadcomm, który jest użyty w komputerze Raspberry Pi 3B. Ze względu na korzystanie z płytki prototypowej oraz modułu Proto Pi Plus, który korzysta z oznaczeń odpowiadających kanałom SOC w projekcie użyto tego sposobu numeracji.

Inicjalizacja obsługi GPIO wywoływana jest w funkcji:

```
def init_gpio():
    GPIO.setmode(GPIO.BCM)
```

Wykonywana jest ona na początku działania programu nadzor.py w funkcji main(). Obsługa poszczególnych czujników odbywa się w ramach obiektów klasy Grupa. Użycie rezystora podciągającego jest wywoływane w programie nadzor.py w konstruktorze obiektu klasy Grupa w następujący sposób.

```
GPIO.setup(self.czujnik.gpio, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

self.czujnik.gpio - to atrybut przechowujący numer wyprowadzenia GPIO. GPIO.IN - konfiguruje wyprowadzenie jako wejście. pull_up_down=GPIO.PUD_UP - to flaga określająca zastosowanie rezystora podciągającego.

0.2 Obsługa czujników podłączonych do magistrali I2C

Wymogiem technicznym systemu jest również obsługa czujników przy użyciu magistrali I2C. W ramach wykonanego systemu została zapewniona obsługa czujników temperatury i wilgotności. Wybrany został czujnik Si7021 firmy Adafruit. Zapewnia on pomiar temperatury w zakresie od -10 do 85°C oraz wilgotności względnej od 0 do 80%. Są to wartości wystarczające w warunkach pracy w zamkniętym pomieszczeniu.

Wymogiem systemu jest również obsługa wielu czujników i kamer. Adres czujników Si 7021 to 0x70. Nie mają one możliwości programistycznej lub sprzętowej zmiany adresu. Oznacza to, że do jednej magistrali mógłby być podłączony bezpośrednio tylko jeden czujnik. Podłączenie większej liczby czujników o tym samym adresie doprowadziłoby do sytuacji, w której nie można jednoznacznie określić, z którego czujnika została odczytany wynik pomairu. Raspberry Pi3 B posiada możliwości obsługi dwóch magistral I2C. W

założeniach projektowych jest wymaganie obsługi do 4 czujników temperatury każdego przypisanego do jednej kamery. Nie jest więc możliwe rozwiązanie polegające na podłączeniu po jednym czujnika do każdej z magistral. Innym rozwiązaniem tej sytuacji jest zastosowanie multipleksera I2C. Multiplekser zadziała jak przełącznik, który pozwoli na zapis i odczyt z jednym czujnikiem naraz. Przykładem takiego urządzenia jest TCA 9548A firmy Adafruit. Pozwala on na podłączenie do 8 urządzeń korzystających z magistrali I2C.

Konfiguracja odbywa się w następujący sposób. Do multipleksera przesyłany jest 8-bitowy kod odpowiadający numerowi kanałowi. Aktywacja kanału oznacza ustawienie bitu o pozycji równej numerowi kanałowi jako 1. Po wysłaniu komendy i znaku STOP kanał jest aktywowany. Kolejne komendy można adresować, używając adresu czujnika temperatury. Pierwszą komendą, którą należy wysłać to komenda pomiaru wilgotności względnej. Dostępne są dwa tryby pomiaru tej wielkości: Hold Master Mode oraz No Hold Master Mode. Pierwszy z nich oznacza, że urządzenie nadrzędne wysyła żądanie pomiaru wilgotności. Urządzenie podrzędne potwierdza otrzymanie żądanie i dokonuje pomiaru. Wymaga to zastosowania rozciągania zegara (clock stretching), które polega na utrzymywania przez urządzenie podrzędne linii zegarowej SCL w stanie niskim. Dzieje się to do momentu zakończenia pomiaru przez urządzenie i wystawieniu jego wyniku do rejestru. Drugi tryb różni się tym, że po potwierdzeniu otrzymania żądania wystawiany jest symbol zakończenia komunikacji. Znając maksymalny czas konwersji pomiaru, urządzenie nadrzędne może odczytać wynik z rejestru po jego zakończeniu.

Do obsługi żądań konieczna była odpowiedni pakiet. Pierwszym zastosowanym pakietem był pakiet python-smbus. Korzysta on ze sterownika wbudowanego w jądro systemu Linux. Została przeprowadzona próba komunikacji w trybie Hold Master Mode ze względu na dostępne komendy protokołu I2C. Zakończyła się ona błędem o kodzie io errno5. Wynikał on z tego, że pakiet smbus obsługuje magistralę I2C zgodnie ze standardem SMBus. Posiada on bardziej ścisłe reguły dotyczące czasu trwania transakcji na magistrali. Przy użyciu pakietu nie było możliwe zastosowanie trybu No Hold Master Mode, ponieważ nie dostarczał on komendy odpowiadającej odczytowi rejestru urządzenia o wskazanym adresie. Wszystkie dostępne komendy zawierały zapis do urządzenia przed odczytem.

Innym pakietem umożliwiającym komunikację poprzez magistralę I2C jest pigpio. Opiera on swoje działanie na bibliotece napisanej w języku C. Przed rozpoczęciem działania z pakietu konieczne jest programu pidpiod. Jest to demon – program działający w tle bez interakcji z użytkownikiem. Pierwszym krokiem jest stworzenie obiektu klasy pigpio.pi. Obiekt ten jest przechowywany jako atrybut klasy Grupa.

Przy użyciu tego modułu możliwe jest przeprowadzenie pomiaru wilgotności względnej w trybie No Hold Master Mode. W pierwszej kolejności wysyłana jest przy pomocy funkcji i2c_write_byte() 8-bitowa komenda pomiaru. Następnie program jest usypiany na 0.05s przy pomocy komendy time.sleep(). Jest to wartość większa niż najdłuższy czas konwersji pomiaru wilgotności względnej podany w karcie katalogowej czujnika Si 7021. Następnie odczytywany jest poprzez funkcję i2c_read_device() wynik pomiaru składający się z dwóch bajtów. Funkcja zwraca liczbę odczytanych bajtów i bajty w formie tablicy. Bajty te można następnie wykorzystać do obliczenia wilgotności względnej na podstawie wzoru z karty katalogowej.

Czujnik Si 7021 dokonuje do pomiaru wilgotności względnej również pomiaru temperatury. Wynik tego pomiaru jest przechowywany w urządzeniu. Można go odczytać, wysyłając komendę o kodzie 0xE0. Całą transakcję można zrealizować przy użyciu jednej komendy i2c_read_i2c_block_data, ponieważ nie ma potrzeby oczekiwania na konwersję pomiaru temperatury. Wysyła ona żądanie wystawienia wyniku pomiaru temperatury.

Następnie odczytywana jest podana liczba bajtów. Bajty te są zapisane do tablicy i użyte do obliczenia wartości temperatury.

0.3 Obsługa kamer USB

Raspberry Pi 3 B posiada 4 gniazda USB, do których mogą podłączone kamery USB. Kamery USB są obsługiwane przy pomocy aplikacji fswebcam. Jest to samodzielny program pozwalający na wywoływanie zdjęć z kamer USB podłączonych do komputerem z systemem typu Unix. Wywołanie tego programu z poziomu skryptu w języku Python wymaga użycia modułu subprocessing. Daje on możliwość otwierania programów w osobnych procesach. Do otwarcia fswebcam użyta została klasa Popen, ponieważ daje możliwość wywołania metody wait(), pozwalającej zaczekać na zakończenie wykonywania się procesu.