Uniwersytet Wrocławski

Wydział Fizyki i Astronomii

Bartłomiej Matyszkowicz

System do testowania i demonstracji regulatora temperatury PID w układzie z wentylatorem i elementem grzejnym

System for PID Controller’s testing and demonstration in a system with a fan and heating element

Praca inżynierska wykonana pod kierunkiem  
 dr inż. Radosława Wasielewskiego  
 w Instytucie Fizyki Doświadczalnej UWr

Wrocław 2018

Streszczenie

[PLACEHOLDER]

Abstract

[PLACEHOLDER]

Spis treści

[1. Wstęp 4](#_Toc1565124)

[2. Cel i założenia pracy 4](#_Toc1565125)

[2.1. Układ pomiarowy 4](#_Toc1565126)

[2.2. Środowisko programistyczne 5](#_Toc1565127)

[3. Implementacja 7](#_Toc1565128)

[3.1. Układ pomiarowy 7](#_Toc1565129)

[3.1.1. Czujnik temperatury 8](#_Toc1565130)

[3.1.2. Element grzejny 10](#_Toc1565131)

[3.1.3. Element chłodzący 10](#_Toc1565132)

[3.2. Środowisko programistyczne 13](#_Toc1565133)

[3.2.1. Arduino IDE 13](#_Toc1565134)

[3.2.2. Python 16](#_Toc1565135)

[4. Testy układu 19](#_Toc1565136)

[5. Pracownia studencka 19](#_Toc1565137)

[5.1. Dobór nastaw 20](#_Toc1565138)

[5.1.1. Strojenie ręczne 20](#_Toc1565139)

[5.1.2. Pierwsza metoda Zieglera-Nicholsa 21](#_Toc1565140)

[5.1.3. Druga metoda Zieglera-Nicholsa 21](#_Toc1565141)

[5.1.4. Metoda Cohena-Coona 21](#_Toc1565142)

[5.2. Przykładowe ćwiczenia 21](#_Toc1565143)

1. Wstęp

[TEXT]

1. Cel i założenia pracy

Głównym zadaniem niniejszej pracy inżynierskiej jest stworzenie prostego układu pomiarowego wraz z dostosowaną do niego aplikacją bazującego na regulatorze PID (proporcjonalno-całkująco-różniczkującym) który umożliwia osobie z niego korzystającej zrozumienie sposobu jego działania w praktyce. W pracy przyjęte zostało założenie, że taki układ mógłby stanowić proste ćwiczenie na pracowni elektronicznej dla studentów, pokazujące m.in. jak bezwładność cieplna wykorzystanego czujnika temperatury może wpływać na czas reakcji regulatora, bądź jakie są wady i zalety takiego regulatora w porównaniu do systemu, którego reakcja na zmianę temperatury następuje dopiero gdy ta przekroczy pewną wartość zdefiniowaną dla takiego systemu. Takie założenie zostało przyjęte ze względu na brak informacji o takim regulatorze pomimo jego szerokiego zastosowania w przemyśle.

* 1. Układ pomiarowy

W założeniach układ pomiarowy, którego zadaniem jest kontrola i regulacja temperatury składa się z trzech głównych komponentów: elementu grzejnego, czujnika temperatury i elementu chłodzącego. W takim układzie jeden element przyjmuje rolę źródła zakłóceń temperatury, zadaniem drugiego jest natomiast jej regulacja na podstawie danych zbieranych przez czujnik temperatury. Innymi słowy taka konstrukcja umożliwia pracę na dwa różne sposoby, zależnie od tego który element przyjmuje rolę źródła zakłóceń. Gdy źródłem zakłóceń jest element grzejny to znaczy, że czujnik temperatury jest stale nagrzewany i zadaniem elementu regulującego jest jego schłodzenie i utrzymanie docelowej temperatury. Natomiast jeśli źródłem zakłóceń jest element chłodzący to celem elementu grzejnego jest dostarczanie odpowiedniej ilości ciepła, aby temperatura układu nie spadła poniżej docelowej.

[PLACEHOLDER – THEORETICAL IMAGE OF SYSTEM]

Taki układ pomiarowy może być sterowany za pomocą logiki regulatora PID. Jest to prosty system jednowymiarowej regulacji – układ steruje pojedyńczym elementem na podstawie danych otrzymywanych z jednego źródła. W przypadku omawianego układu źródłem danych jest czujnik temperatury, element sterowany jest zależny od wybranego sposobu pracy układu.

Czas reakcji i precyzja regulatora w pełni zależy od 3 parametrów – tzw. nastaw. Są to kolejno składowa proporcjonalna, całkująca i różniczkująca. Każdy parametr spełnia inne zadanie. Składowa proporcjonalna reguluje czas reakcji regulatora na zmianę temperatury. Zadaniem składowej całkującej jest sprowadzenie różnicy między obecną temperaturą a punktem docelowym do zera. W tym celu jeśli ta różnica przez pewien czas będzie większa niż zero, to składowa będzie dodawać do sygnału sterującego pewną wartość zależną od różnicy. Składowa różniczkująca ma na celu wytłumienie oscylacji temperatury wokół zadanego progu. Odpowiedni dobór tych trzech parametrów pozwala na stabilną pracę układu i jest podstawowym problemem każdego regulatora PID.

* 1. Środowisko programistyczne

Regulacja i kontrola regulatora powinna odbywać się z poziomu aplikacji komputerowej umożliwiającej użytkownikowi odpowiedni dobór nastaw regulatora oraz temperatury docelowej układu. Zadaniem aplikacji jest też pobranie danych z czujnika temperatury i przedstawienie ich na wykresie, co powinno pozwolić na szybką analizę poprawności dobranych parametrów na podstawie zachowania temperatury.

Jedną z zakładanych opcji aplikacji jest możliwość przełączenia układu z regulacji PID na pracę w trybie prostego przełącznika typu ON/OFF. W takim trybie regulator włączałby chłodzenie po przekroczeniu temperatury docelowej i wyłączał je gdy ta spadnie poniżej zadanego poziomu. Udostępnienie takiego trybu pracy pozwoli użytkownikowi na porównanie różnic między zachowaniem regulatora i wachaniami temperatury przy różnych sposobach regulacji, oraz na ocenienie w jakich sytuacjach prostszy regulator może być przydatniejszy od wymagającego regulacji kontrolera PID.

Sama aplikacja nie powinna bezpośrednio wpływać na to w jaki sposób zachowywać się będzie układ – cel ten spełnia system wbudowany sterowany z poziomu aplikacji. Aplikacja powinna przesyłać do niego dane takie jak temperatura docelowa układu oraz nastawy PID, zadaniem systemu wbudowanego jest natomiast na podstawie tych danych oraz temperatury odebranej z czujnika wyznaczenie odpowiednich parametrów pracy sterowanych komponentów w taki sposób, aby temperatura układu była stabilna. Aby system mógł spełniać swoje zadania musi być on w stanie odbieraći wysyłać sygnały by być w stanie sterować elementami i odczytywać dane z czujnika oraz musi mieć możliwość przeprowadzania obliczeń do wyznaczania parametrów pracy sterowanych układów. Spełnienie tych warunków pozwala systemowi wbudowanemu na bycie łącznikiem pomiędzy aplikacją sterowaną przez użytkownika a układem pomiarowym.

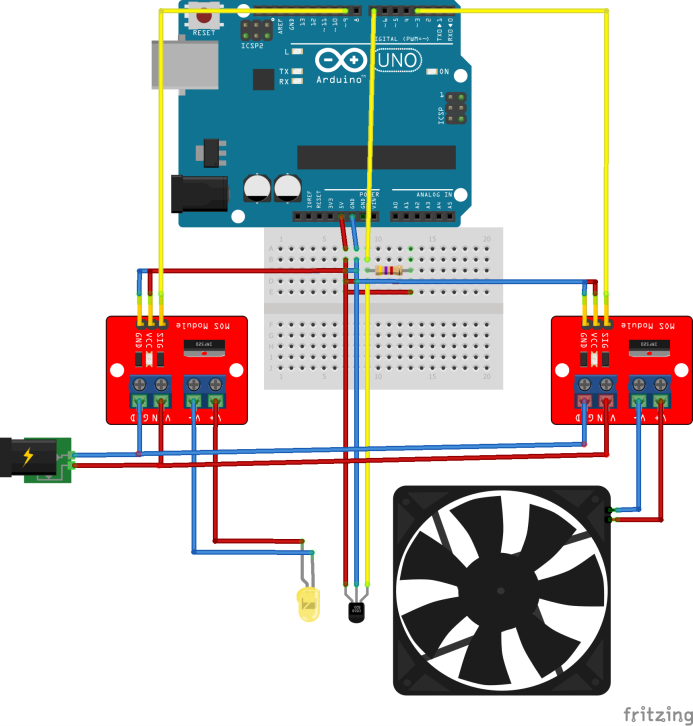
[PLACEHOLDER – BLOKOWY SCHEMAT AKWIZYCJI I KONTROLI – ZDJĘCIE JEST NA TELEFONIE]

1. Implementacja
   1. Układ pomiarowy

Ze względu na niewielką liczbę elementów potrzebnych do zbudowania układu, do jego stworzenia wykorzystany został mikrokontroler Arduino Uno w rewizji 1. Do podłączenia wszystkich komponentów potrzebne są 2 cyfrowe wyjścia do sterowania mocą żarówki i wentylatora oraz jedno cyfrowe wejście do odbioru danych z czujnika temperatury. Arduino Uno udostępnia 14 cyfrowych wejść/wyjść, z czego 6 jest możliwych do wykorzystania jako kanały PWM – jeden jest potrzebny do odpowiedniego sterowania prędkością wentylatora.

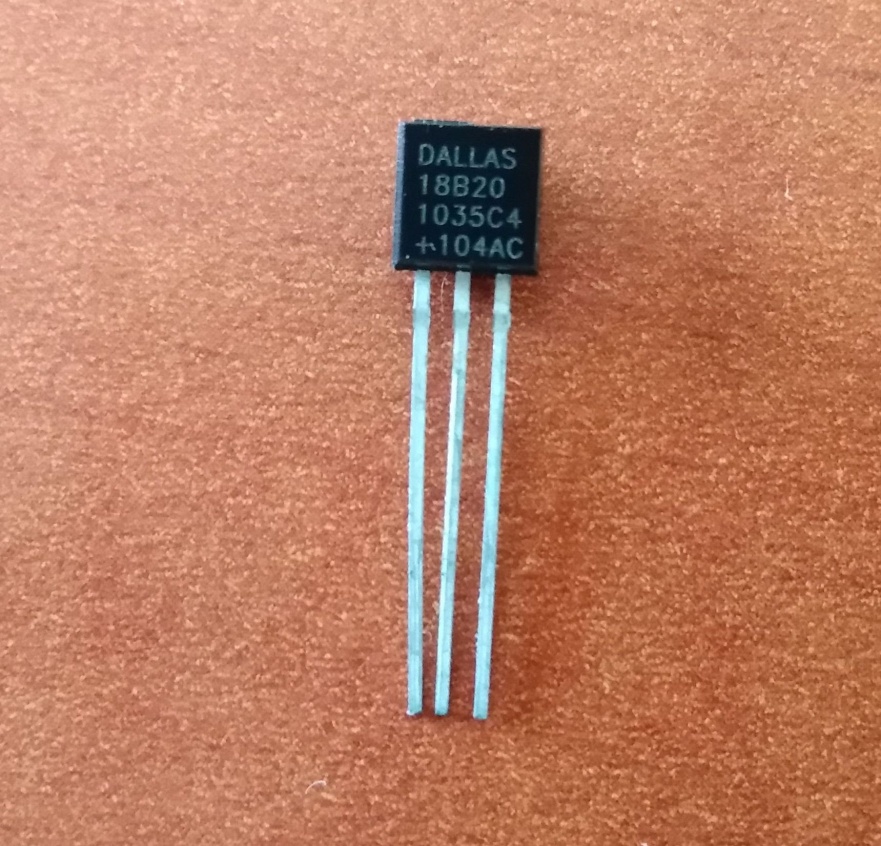


Do wymiany informacji między mikrokontrolerem a aplikacją komputerową użyty został interfejs USB typu B dostępny na płytce Arduino Uno. Jego zadaniem jest wysłanie informacji odebranych z czujnika temperatury do aplikacji oraz odebranie wartości sygnału PWM za pomocą którego następuje sterowanie prędkością wentylatora. Warty uwagi jest fakt, że interfejs USB jest magistralą szeregową – dane są wysyłane pod postacią jednego ciągu bitów, który musi dotrzeć do odbiorcy zanim następny pakiet danych zostanie wysłany. Wiąże się to z koniecznością chwilowego zatrzymania wysyłania informacji o mierzonej temperaturze w celu odebrania sygnałów sterujących z aplikacji. Więcej na ten temat opisane zostanie w podrozdziale 3.2.



* + 1. Czujnik temperatury

Do pomiaru temperatury użyty został czujnik temperatury DS18B20 pracujący na zasadzie interfejsu 1-wire. Jest to zarówno interfejs elektroniczy oraz protokół komunikacyjny między urządzeniami komunikującymi się między sobą korzystając z pojedyńczej szyny danych. Jedną z zalet wykorzystania takiego interfejsu jest fakt, że odbiornik (w przypadku układu przedstawionego w pracy jest to czujnik temperatury) jest zasilany za pomocą szyny danych – pozwala to na proste podłączenie takiego odbiornika bez konieczności podłączania go do zasilania dodatkowymi przewodami, sprawiając, że układ jest bardziej przejrzysty oraz prostszy w modyfikacji i utrzymaniu.

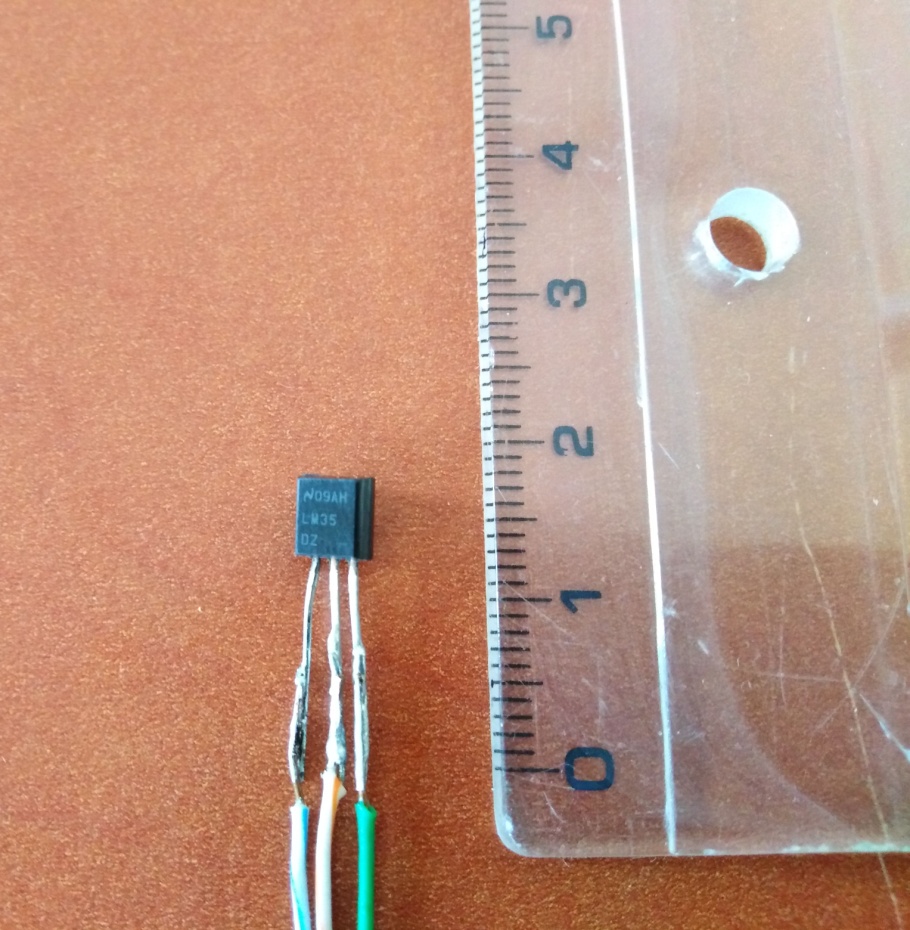


Czujnik DS18B20 pracuje w zakresie od -55˚C do 125˚C z dokładnością ±0.5˚C w zakresie -10˚C do 85˚C. Zakresy te w zupełności powinny wystarczyć przy użytkowaniu układu korzystającego z żarówki halogenowej o mocy nieprzekraczającej 35 watów.

Alternatywnym czujnikiem, który mógłby zostać wykorzystany w układzie jest LM35. Pracuje on w zakresie temperatur od 0 do 100 stopni Celsjusza. W przeciwieństwie do wykorzystanego sensora DS18B20, LM35 jest czujnikiem analogowym, co oznacza, że wartość podawana na wyjściu jest wartością napięcia. LM35 jest czujnikiem liniowym o regulacji 10 mV/⁰C, co oznacza, że przy zmianie temperatury o 1⁰C napięcie wyjściowe wzrasta o 10 mV.

Sposób w jaki czujnik LM35 przekazuje temperaturę do kontrolera sprawia, że jej odczytanie jest bardzo proste: odczytaną wartość należy przemnożyć przez wartość napięcia referencyjnego (w przypadku użytego kontrolera Arduino jest to 5 V), wynik podzielić przez rozdzielczość kontrolera, na koniec zaś trzeba przeliczyć wartość z miliwoltów na wolty mnożąc wynik przez 100.

Prostota odczytania temperatury z czujnika LM35 w porównaniu do DS18B20, który wymaga użycia dostosowanej biblioteki jest dużą zaletą przy rozważaniach który czujnik powinien zostać użyty w układzie. Użycie DS18B20 jest jednak uzasadnione możliwością rozbudowy układu o kolejne czujniki umożliwiające zwiększenie dokładności pomiaru – interfejs 1-Wire pozwala na podłączenie kilku sensorów na jednej szynie danych, co nie jest możliwe dla zwyczajnych czujników analogowych. Chcąc rozbudować układ o kolejne LM35 konieczne jest zastosowanie większej ilości przewodów, które ze względu na położenie czujnika wewnątrz układu mogą zakłócać przepływ temperatury.



* + 1. Element grzejny

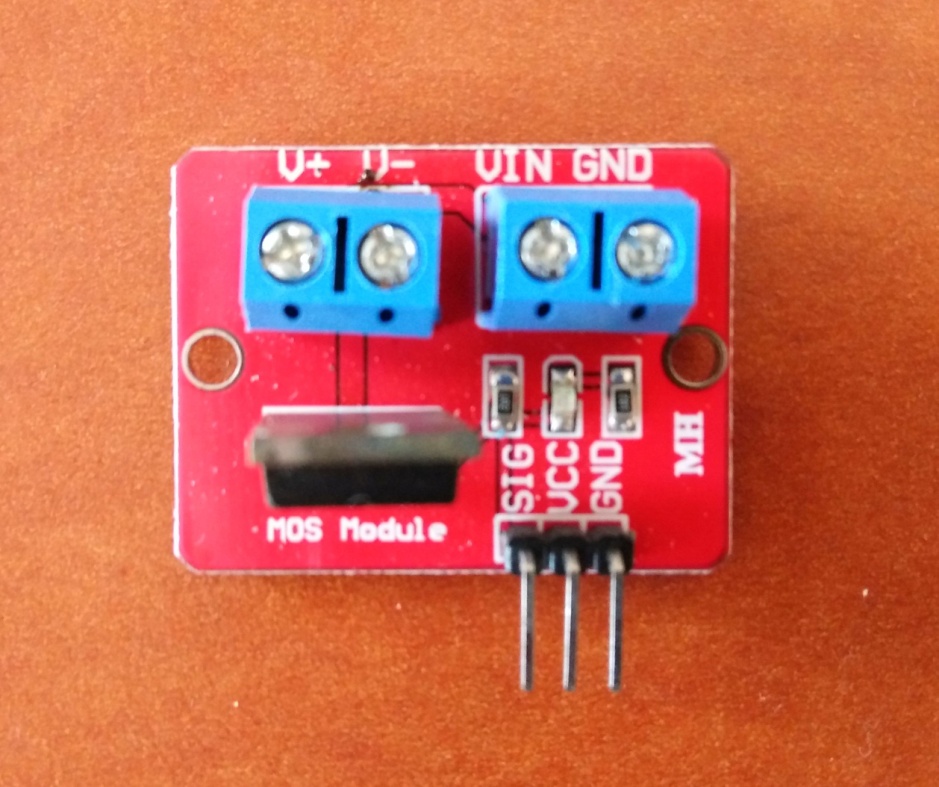
Do symulowania wzrostu temperatury wewnątrz układu niezbędne jest zastosowanie elementu, który będzie wydzielał ciepło nagrzewające czujnik. W tym celu wykorzystana została żarówka halogenowa Decostar 35 12V o mocy 35 watów. Jest ona przymocowana na prowadnicy kulkowej umieszczonej wewnątrz tuby, pozwalając na manualną regulację odległości żarówki od czujnika temperatury. Odległość żarówki od czujnika wpływa na szybkość przyrostu notowanej temperatury – im żarówka jest dalej tym wolniej czujnik będzie notował zmianę.

* + 1. Element chłodzący

Jako element chłodzący w układzie wykorzystany został wentylator Protechnic Magic o wymiarach 80x80mm zasilany prądem o natężeniu 12V i napięciu 0.15A. Rozmiar wentylatora jest niewielki, co umożliwiło wykorzystanie mniejszej niż początkowo planowano tuby ze szkła akrylowego, zmniejszając ogólny rozmiar układu, doprowadziło to jednak do nieoczekiwanego wcześniej problemu – gdy żarówka jest w minimalnej odległości od czujnika taki wentylator, nawet pracując z maksymalną mocą, nie jest w stanie odpowiednio schłodzić szybko nagrzewającego się czujnika aby sprowadzić jego temperaturę do punktu docelowego regulatora. Rozwiązania takiego problemu są trzy: można wykorzystać wentylator o większej mocy lub większego rozmiaru, wiązałoby się to jednak z koniecznością przebudowania całego układu, ponieważ do pełnego wykorzystania możliwości wentylatora o większej powierzchni chłodzącej należałoby wykorzystać tubę o odpowiednio większej średnicy. Innym rozwiązaniem problemu jest zwyczajne zwiększenie minimalnej odległości do czujnika w jakiej można ustawić żarówkę. Można także wykorzystać żarówkę o mniejszej mocy, prowadzi to jednak do pewnego wydłużenia czasu strojenia regulatora ze względu na dłuższy czas nagrzewania się czujnika.



Do sterowania mocą wentylatora wykorzystany został moduł wykonawczy MOSFET IRF520. Jest to układ wykonawczy z tranzystorem mocy umożliwiający sterowanie urządzeniami o wyższym poborze prądu niż maksymalny dopuszczalny przez płytkę sterującą – w przypadku Arduino Uno jest to 12V. Układ IRF520 pozwala na wykorzystanie napięcia zasilania w zakresie 0-24 V pobierając prąd do 5 A. Jest to bardzo przydatna cecha, ponieważ umożliwia ona rozbudowę układu o mocniejsze elementy grzejne i chłodzące w przypadku gdyby zaszła taka konieczność, nie jest jednak głównym powodem użycia takich modułów. Jest nim prostota z jaką można sterować mocą przekazywaną na sterowany element.



Moduł MOSFET posiada 7 wyprowadzeń, z czego 3 pary z nich są złączami zasilania – jedna odpowiada za podłączenie zasilania z modułu sterującego, druga za podłączenie zasilania z zewnętrznego źródła natomiast trzecia służy do podłączenia przewodów zasilających sterowany element. Ostatnie wyprowadzenie znajduje się pomiędzy złączami zasilania z płytki sterującej i służy ono do odbioru sygnału sterującego elementem. Zastosowanie takiego modułu pozwala zatem na zredukowanie liczby przewodów i rezystorów koniecznych do obsługi sterowania elementem.

Moduł MOSFET IRF520 pozwolił także na rozwiązanie drobnego problemu powstałego przy zasilaniu i sterowaniu wentylatorem. Gdy ten korzystał ze zwykłego tranzystora MOSFET konieczne było podłączenie dodatkowej diody Schottky’ego między zasilaniem wentylatora a uziemieniem, która służyła do rozładowania wentylatora gdy ten powinien się wyłączyć po otrzymaniu zerowego sygnału PWM. Tak podłączona dioda pracuje jako dioda rozładowcza – wentylator zachowuje się niczym ładowana cewka i po odcięciu zasilania pracuje on dalej, zadaniem diody jest natomiast jego natychmiastowe rozładowanie. Zastosowanie modułu IRF520 pozwoliło na dalsze uproszczenie układu, ponieważ moduł posiada diodę rozładowczą w swojej konstrukcji.

Taki sam moduł został również wykorzystany do podłączenia elementu grzewczego. Nie jest on konieczny przy standardowym trybie pracy, polegającym na chłodzeniu nagrzewającego się elementu, jego wykorzystanie umożliwia jednak wprowadzenie dodatkowej koncepcji, według której układ może pracować – jeśli element grzejny stanie się elementem kontrolowanym przez układ, zaś element chłodzący będzie pracował ze stałą, maksymalną mocą to zadaniem układu przestaje być chłodzenie czujnika, by ten nie osiągał temperatur wyższych niż zadane. Jego nowym zadaniem jest nagrzanie czujnika do temperatury docelowej i utrzymanie tej temperatury na zadanym progu pomimo chłodzenia. Zmiana warunków pracy regulatora pozwala zarówno na odmienne spojrzenie na sposób kontroli temperatury oraz umożliwia przeprowadzenie doświadczenia mającego za cel wykazanie która metoda pozwala na skuteczniejsze utrzymanie temperatury na zadanym poziomie bez odchyłów w obu kierunkach.

* 1. Środowisko programistyczne
     1. Arduino IDE

Podstawowym elementem układu pomiarowego jest platforma sprzętowa Arduino pełniąca rolę regulatora PID. Do pełnienia swoich funkcji korzysta ona z trzech bibliotek umożliwiających jej kontakt z wykorzystywanymi urządzeniami. Biblioteki „OneWire” i „DS18B20” służą do nawiązania połączenia z czujnikiem temperatury DS18B20, natomiast biblioteka „PID\_v1” udostępnia różne funkcje umożliwiające stworzenia regulatora PID.

Kod wgrany na platformę Arduino napisany został w środowisku Arduino IDE, udostępnionym przez producenta tejże platformy. Kod wykorzystywany przez platformę jest zbiorem funkcji w językach C i C++, które są wywoływane przez platformę. Głównym zadaniem, jakie wykonuje jest obsługa sygnałów przychodzących z aplikacji i regulacja pracy regulatora PID.

Kod można podzielić na trzy części: importowanie bibliotek wykorzystywanych przez program, funkcje systemowe niezbędne do działania kodu oraz funkcje użytkownika.

Zewnętrzne biblioteki służą do rozszerzenia możliwości oferowanych przez platformę Arduino bądź do wprowadzenia zupełnie nowych funkcjonalności. Kod wykorzystywany przez platformę użytą w układzie prezentowanym w pracy importuje trzy takie biblioteki: DS18B20, OneWire oraz PID\_v1. Biblioteka OneWire służy do obsługi protokołu OneWire poprzez stworzenie obiektu, za pomocą którego należy odwoływać się do czujnika temperatury z którego pobierany jest wynik pomiaru. Ponieważ interfejs OneWire umożliwia podłączenie kilku czujników temperatury wysyłających dane na tej samej magistrali, do poprawnego działania potrzebuje 64-bitowego numeru seryjnego. Biblioteka OneWire udostępnia funkcje umożliwiające odczytanie numeru seryjnego czujnika podłączonego do określonego wejścia/wyjścia płytki Arduino. Odczytany numer należy przekazać odpowiedniemu obiektowi, aby ten stworzył połączenie na interfejsie OneWire z używanym czujnikiem.

Następnie, za pomocą biblioteki DS18B20 definiuje się obiekt bezpośrednio reprezentujący sensor, który korzysta z referencji do stworzonego połączenia typu OneWire. Biblioteka DS18B20 oferuje szereg funkcji do obsługi podłączonego czujnika, z czego w kodzie obsługującym regulator wykorzystane zostały trzy: funkcja ***Begin()***, służąca do uruchomienia czujnika i zdefiniowania jego parametrów pracy takich jak rozdzielczość, funkcja ***readTemperature()***pobierająca z czujnika temperaturę oraz funkcja ***request()***, wysyłająca do czujnika polecenie wykonania pomiaru temperatury.

Do implementacji funkcjonalności regulatora PID użyta została biblioteka PID\_v1 pozwalająca stworzyć obiekt reprezentujący regulator. Do jego utworzenia potrzebna jest temperatura docelowa jaka będzie utrzymywana w układzie, obecna temperatura, adres na zmienną, która będzie przyjmowała sygnał PWM wyliczony tak, aby podana temperatura została utrzymana, wartości trzech nastaw regulatora oraz „kierunek działania” regulatora – wartość ta definiuje w jaki sposób regulator wylicza sygnał PWM wysyłany na sterowany element. Przy otrzymaniu polecenia DIRECT sygnał będzie zmniejszany gdy obserwowana wartość osiągnie docelową, polecenie REVERSE natomiast będzie ten sygnał zwiększać. Polecenia te można przyrównać do sposobu prowadzenia samochodu i chłodzenia lodówki – w przypadku prowadzenia samochodu jeśli dozwolona prędkość zostanie przekroczona należy zacząć ją zmniejszać, natomiast w przypadku lodówki jeśli temperatura zacznie przekraczać docelową należy mocniej chłodzić. Na potrzeby regulatora wykorzystany jest tryb pracy REVERSE.

Biblioteka PID udostępnia szereg funkcji do sterowania regulatorem. Do poprawnego działania wykorzystane są dwie z nich – funkcje ***SetTunings()*** oraz ***Compute()***. Pierwsza z nich otrzymuje jako parametry nowe nastawy z których powinien korzystać regulator i pozwala na ich zmianę w trakcie działania programu, natomiast druga funkcja służy do przeprowadzenia obliczeń, które prowadzą do wyznaczenia odpowiedniego sygnału PWM. Metoda obliczeń jest bardzo prosta – najpierw wyznaczane są różnice między wartością docelową i obecną oraz obecną i poprzednią wartością, które następnie wykorzystane są do wyliczenia trzech składowych sygnału docelowego. Składowa proporcjonalna jest iloczynem nastawy proporcjonalnej z różnicą między obecną i poprzednią wartością, składowa całkująca jest iloczynem nastawy całkującej z różnicą między wartością docelową a obecną, natomiast składowa różniczkująca jest wyznaczona podobnie do składowej proporcjonalnej, z wykorzystaniem nastawy różniczkującej. Trzy składowe są sumowane i w razie konieczności ograniczone do zakresu 0-255. Wyznaczony sygnał jest przypisany do zmiennej, której adres został podany w konstruktorze obiektu.

Kod, z którego ma korzystać płytka Arduino powinien posiadać dwie funkcje systemowe, pozwalające jej na sprawne działanie. Pierwszą z nich jest uruchamiana jednorazowo przy uruchomieniu płytki funkcja ***setup()***, której zadaniem jest przygotowanie platformy do działania poprzez inicjalizację portów wejścia/wyjścia oraz otwarcie portu szeregowego za pomocą którego odbywa się komunikacja płytki z komputerem. Ponadto, na potrzeby funkcjonowania regulatora PID, funkcja uruchamia jego obiekt utworzony z biblioteki PID\_v1.

Funkcja ***loop()***, uruchamiana po wykonaniu wszystkich instrukcji funkcji ***setup()***, jest główną pętlą programu, działającą nieprzerwanie do momentu wyłączenia płytki. W każdej iteracji pętli następuje wywołanie funkcji sczytującej dane z komputera, oraz zależnie od tego czy zostały odebrane nowe dane, aktualizowane są nastawy regulatora. Następnie funkcja sprawdza jaką komendę otrzymał regulator i na jej podstawie wywołuje odpowiednie funkcje kontrolera temperatury.

Zasada działania kontrolera temperatury oparta jest na prostej maszynie stanów zbudowanej na instrukcji warunkowej switch. W każdej iteracji głównej pętli sterującej Arduino program sprawdza w jakim stanie powinien pracować po czym wywołuje odpowiednią funkcję. Wybór stanu odbywa się z poziomu aplikacji komputerowej przy wykorzystaniu przycisków uruchamiających bądź wyłączających stany z nimi powiązane. Zastosowanie takiego wzorca projektowego uzasadnione jest prostotą, z jaką powinien zachowywać się układ – w danym momencie wykonuje on tylko jedną, konkretną funkcję, której zakres nie wykracza poza pojedyńczy stan. Ponadto umożliwia on w razie konieczności szybką i łatwą implementację dodatkowych stanów, bez konieczności modyfikacji kodu istniejących już istniejących.

Podstawowym stanem w jakim pracuje układ jest oczekiwanie – będąc w nim program stale wysyła do aplikacji informacje o zbieranej temperaturze oraz utrzymuje wentylator i lampę halogenową w stanie bezczynności poprzez nieustanne nadawanie zerowego sygnału PWM na oba elementy.

Drugim stanem regulatora jest stan prostej regulacji temperatury. Zasada jego działania bazuje na nieskomplikowanym przełączniku typu ON-OFF. Pracując w nieskończonej pętli kontroluje on co 100 milisekund stan temperatury pobranej z czujnika i porównuje ją z zadaną temperaturą jaką powinien utrzymać. Sterowanie elementem chłodzącym odbywa się na zasadzie regulacji dwupołożeniowej sterowanej z poziomu aplikacji komputerowej – regulator otrzymuje pewną wartość kontrolną, która jest następnie wykorzystana do wyznaczenia symetrycznego zakresu wokół temperatury docelowej. Element sterowany zostaje uruchomiony gdy temperatura przekroczy górny zakres lub wyłączony gdy spadnie poniżej dolnego. Taka mechanika pozwala na uniknięcie sytuacji gdy temperatura zaczyna oscylować wokół progu docelowego z niewielką amplitudą, powodując ciągłe włączanie i wyłączanie sterowanego elementu w minimalnych odstępach czasowych.

Na sam koniec iteracji pętli temperatura zostaje wysłana do aplikacji w celu zaktualizowania wykresu. Przerwanie tej pętli odbywa się za pomocą przycisku sterującego w aplikacji i powoduje powrót do stanu oczekiwania.

Najważniejszym stanem w jakim może pracować układ jest stan regulatora PID. Sposób jego działania jest bardzo zbliżony do stanu prostej regulacji, jednak zamiast zwykłego porównania temperatur wywołuje on na obiekcie kontrolera stworzonym w funkcji systemowej setup funkcję obliczającą sygnał, jaki powinien zostać wysłany do wentylatora. Wartość sygnału jest sumą trzech składowych wyliczonych na podstawie nastaw otrzymanych z aplikacji.

Ostatnim trybem pracy regulatora jest stan chłodzenia czujnika. Będąc w nim regulator operując w nieskończonej pętli utrzymuje lampę halogenową w bezczynności i wymusza na wentylatorze pracę z maksymalną mocą do momentu otrzymania sygnału przerwania. Tryb ten został zaimplementowany w celu przyspieszenia okresu oczekiwania pomiędzy skończeniem pomiaru dla jednego zestawu nastaw i temperatury docelowej a rozpoczęciem kolejnego, umożliwiając schłodzenie czujnika do temperatury w jakiej rozpoczynany był poprzedni pomiar.

* + 1. Python

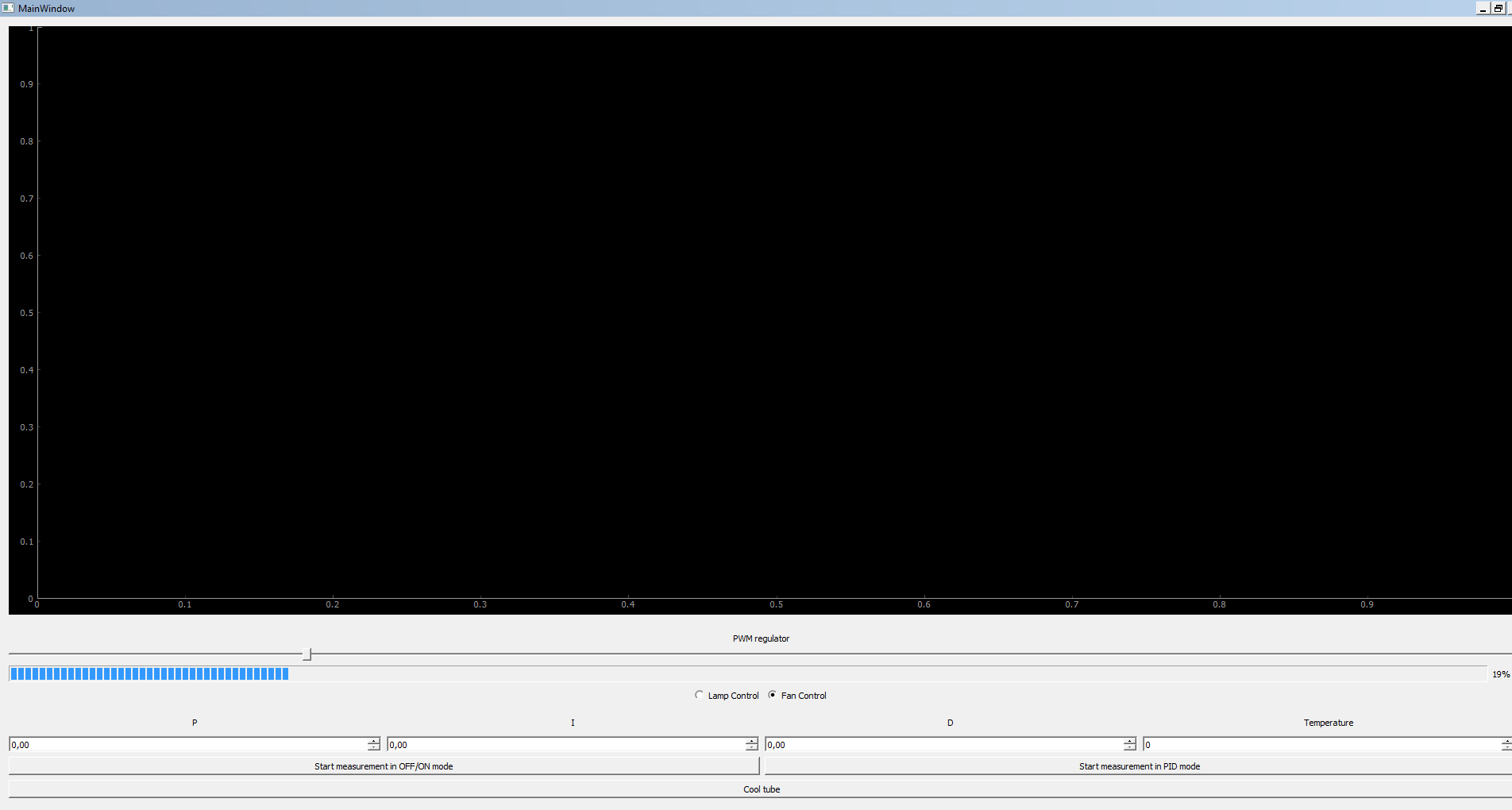
Główna aplikacja służąca do sterowania układem została napisana w języku skryptowym Python. Wybór ten podyktowany jest szerokim asortymentem różnorodnych bibliotek możliwych do zaimportowania przy pisaniu kodu, które m.in. ułatwiają komunikację z platformą Arduino bądź umożliwiają i znacząco upraszczają rysowanie wykresów na podstawie otrzymywanych danych. Biorąc pod uwagę konieczność odebrania danych z układu pomiarowego i wysyłania ich do niego, najważniejszą wykorzystaną biblioteką jest „pySerial” umożliwiająca nawiązanie połączenia z portem szeregowym obecnie używanym przez Arduino za pomocą jednej linii kodu. Biblioteka „PyQt5” służy do połączenia funkcji programu z graficznym interfejsem użytkownika GUI stworzonym w w programie QtCreator, natomiast biblioteka „PyQtGraph” jest użyta do połączenia się z polem stworzonym w oknie GUI i narysowania w nim wykresu na bazie otrzymanych danych.

Do komunikacji pomiędzy Arduino a aplikacją komputerową konieczne jest użycie biblioteki serial. W tym celu należy stworzyć obiekt połączenia szeregowego, który będzie używany przez program do odbierania i wysyłania danych. Gdy obiekt jest już utworzony pozwala on na wysyłanie danych na port za pomocą funkcji ***write()*** oraz na ich odbiór przy wykorzystaniu funkcji ***read()*** służącej do sczytania pojedynczego znaku bądź funkcji ***readline()***, czytającej całą linię. Warto zaznaczyć, że port szeregowy przesyła dane pod postacią pojedynczych bitów – oznacza to, że w celu wysłania bądź odebrania ciągu znaków należy je zakodować lub odkodować funkcjami ***encode()*** oraz ***decode()***. Jeżeli odbierane dane nie będą dekodowane, to zamiast oczekiwanego tekstu bądź wartości zapisane zostaną wartości ASCII odpowiadające przesyłanym symbolom. W przeciwieństwie do Arduino nie istnieje jednak konieczność odbierania każdego znaku z osobna i zapisywania go do tymczasowego buforu, który zostanie scalony z pozostałymi znakami – funkcja ***readline()*** robi to za użytkownika, zapisując do zmiennej cały ciąg znaków zakończony symbolem końca linii.

Działanie aplikacji opiera się na oknie z interfejsem graficznym posiadającym wszystkie elementy służące do sterowania i obserwacji zachowania regulatora. W jego skład wchodzą: wykres temperatury, regulacja poziomu wyzwalania PWM z paskiem informującym o jego stanie, 2 pola typu radio button służące do wyboru elementu regulowanego przez układ, 4 spinnery służące do nadania wartości nastaw regulatora oraz docelowej temperatury, 2 przyciski do włączenia pomiarów w dwóch trybach – regulatora typu ON/OFF oraz regulatora PID i przycisk do schłodzenia układu. Regulacja poziomu wyzwalania PWM pozwala na wyznaczenie wartości odcięcia - gdy sygnał nadawany na sterowany element będzie mniejszy od tej wartości zostanie on zastąpiony zerem. Wymuszenie takiego zachowania na elemencie sterowanym podyktowane jest zastosowaniem wentylatora jako podstawowego elementu chłodzącego – w teorii wentylator powinien rozpocząć pracę nawet gdy dostanie najmniejszy możliwy sygnał – 1, jednak w praktyce taki sygnał jest niewystarczający do pokonania oporów wewnętrznych - wentylator pomimo dostarczanej mocy nie jest w stanie wykonać żadnej pracy, zaczyna się nagrzewać, co może prowadzić do jego uszkodzenia.

Od strony programistycznej okno jest klasą inicjalizującą wszystkie elementy interfejsu i w przypadku przycisków podłączającą funkcje mające się wykonać w momencie ich naciśnięcia. Funkcje te można podzielić w zależności od tego do jakiego typu przycisku są podłączone. Zadanie funkcji podłączonych do przycisków typu radio jest bardzo proste – zmieniają one wartość zmiennej odpowiadającej za to, którym elementem powinien sterować układ. Jest ona następnie wykorzystana w bardziej rozbudowanych funkcjach podłączonych do zwyczajnych przycisków typu ON/OFF. Podstawową z trzech takich funkcji jest funkcja odpowiadająca za uruchomienie układu w trybie regulatora PID. Sczytuje ona ze wszystkich spinnerów wartości nastaw i temperatury docelowej, po czym scala je razem ze zmienną kontroli elementu i komendą rozpoczęcia pomiaru w jeden ciąg znaków i wysyła zakodowany na podłączony port szeregowy. Funkcja służąca do uruchomienia regulatora w trybie ON/OFF jest jej uproszczoną wersją – ponieważ regulacja ON/OFF nie wymaga żadnego parametru poza temperaturą docelową, sczytana jest tylko ona, natomiast wysłane nastawy są ustawione na wartość zerową. Pomimo tego, że nastawy te nie są wykorzystane w tym trybie muszą one zostać wysłane ze względu na sposób w jaki funkcja służąca do odbioru danych po stronie Arduino je odczytuje i zapisuje – kolejność parametrów w ciągu znaków jest istotna i pominięcie chociaż jednego z nich może prowadzić do zapisania nieodpowiednich wartości w zmiennych używanych przez płytkę Arduino.

Trzecia funkcja, służąca do schłodzenia układu jest najprostszą ze wszystkich – odpowiada jedynie za nadanie odpowiedniej komendy sterującej, pozostałe parametry, podobnie jak w funkcji regulatora ON/OFF są wysłane jako zera.



[PLACEHOLDER – OPISAĆ RYNEK – NATIONAL INSTRUMENTS MA COŚ PODOBNEGO]

[PLACEHOLDER – JAKO ALTERNATYWY DS18B20 OPISAĆ TERMISTOR]

1. Testy układu

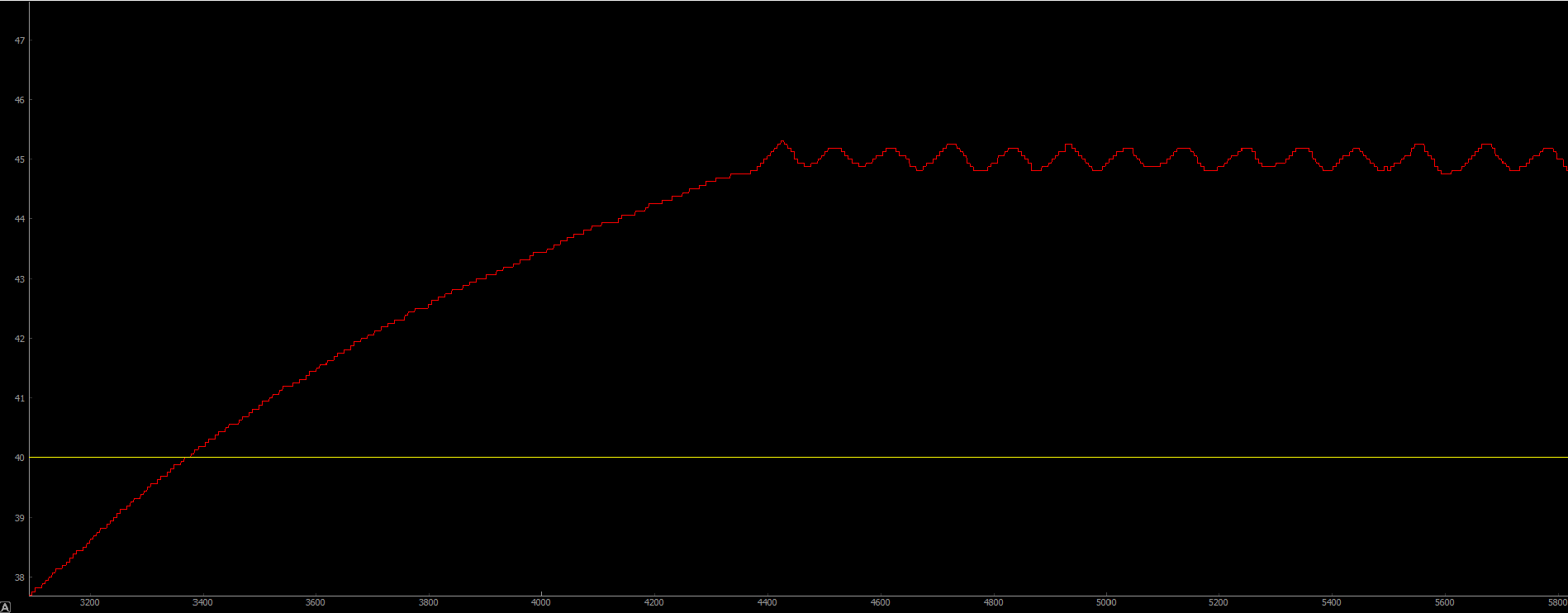
[PLACEHOLDER]

1. Pracownia studencka

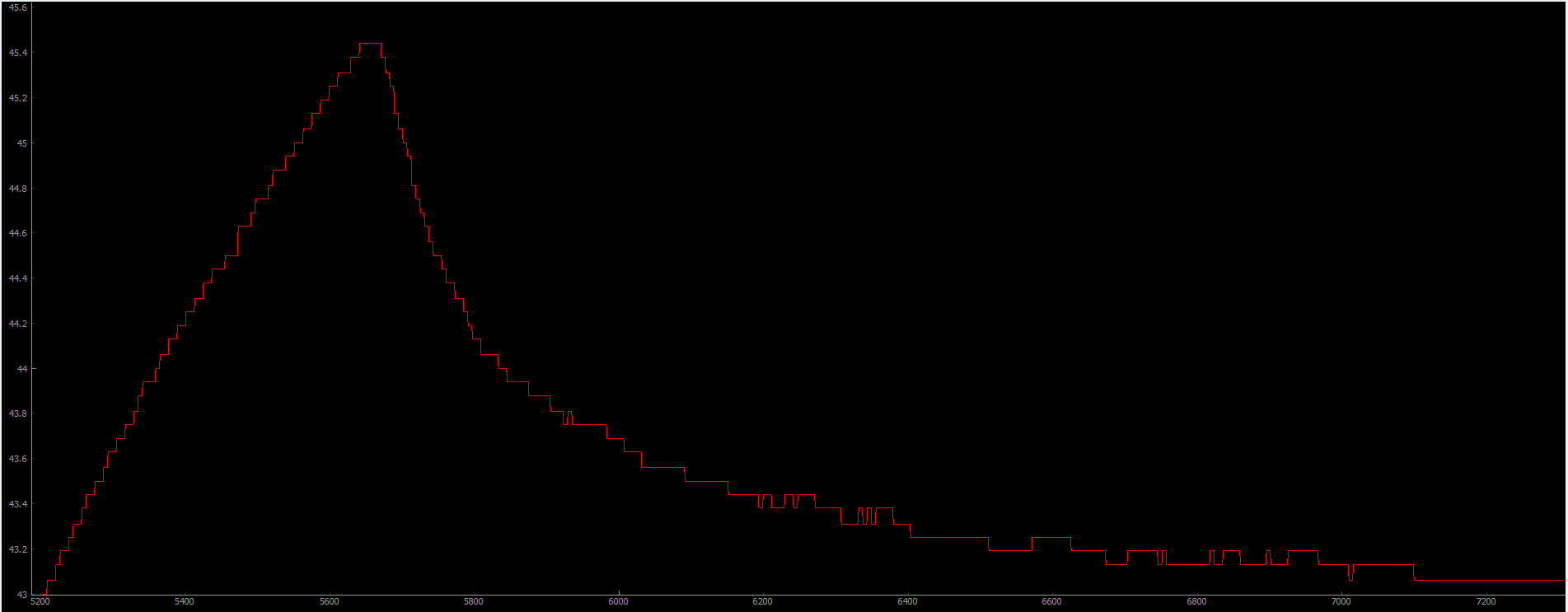
Jednym z podstawowych założeń pracy jest dostosowanie układu w taki sposób, aby możliwe było jego użycie na pracowni studenckiej. Zachowanie układu zależne jest od trzech zmiennych warunków: doboru nastaw PID, odległości elementu grzejnego od czujnika oraz trybu pracy dwóch elementów sterowalnych układu. Bazując na różnych kombinacjach tych zmiennych możliwe jest stworzenie ćwiczeń wymagających od studenta np. dobrania odpowiednich parametrów i trybu pracy celem jak najszybszego ustabilizowania temperatury na zadanym poziomie bądź porównania różnych trybów pracy w stałych warunkach.

* 1. Dobór nastaw
     1. Strojenie ręczne

Metoda strojenia ręcznego polega na doświadczalnym dostosowaniu nastaw bazując na zachowaniu kontrolera w danej chwili. W pierwszym kroku resetuje się wszystkie trzy nastawy. Następnie należy stopniowo zwiększać składową proporcjonalną (nastawę P) do momentu, w którym temperatura badanego układu nie wejdzie w stan regularnych oscylacji o zbliżonej sobie amplitudzie.



Gdy zachowanie układu będzie zadowalające rozpoczyna się powolne zwiększanie składowej różniczkującej (nastawę D). Sprawi to, że oscylacje układu zaczną być stopniowo tłumione. Zwiększanie nastawy D kontynuuje się tak długo, aż oscylacje nie zostaną całkowicie wygaszone. Przy tak dobranych nastawach P i D wykres temperatury badanego układu powinien zachować podobną charakterystykę do przedstawionego:



Sposób w jaki zachowuje się układ przy takich parametrach jest zbliżony do końcowego efektu jaki należy osiągnąć, jednak czas w jakim układ będzie reagował na zmianę temperatury niekoniecznie jest satysfakcjonujący. Aby go skrócić można powtarzać drugi i trzeci krok strojenia regulatora – najpierw następuje zwiększenie składowej proporcjonalnej do uzyskania stabilnych oscylacji, następnie zwiększa się składową różniczkującą do momentu ich wytłumienia. Takie strojenie jest możliwe do momentu gdy zwiększanie nastawy D przestanie tłumić oscylacje.

Ostatnim krokiem potrzebnym do nastrojenia układu jest uregulowanie składowej całkującej, czyli nastawy I. Zwiększa się ją tak długo jak błąd między temperaturą utrzymywaną przez układ a temperaturą docelową nie będzie równy zeru.

* + 1. Pierwsza metoda Zieglera-Nicholsa

[PLACEHOLDER]

* + 1. Druga metoda Zieglera-Nicholsa

Druga metoda Zieglera-Nicholsa jest metodą heurystyczną polegającą na matematycznym wyznaczeniu nastaw I i D na podstawie okresu oscylacji układu. W celu ich wyznaczenia należy wszystkim nastawom nadać wartość zerową, po czym zacząć drobnymi krokami zwiększać nastawę proporcjonalną do najwyższej możliwej wartości przy której występować będą stabilne, niegasnące oscylacje. Następnie odczytuje się okres oscylacji *Pu* oraz zapisuje się wartość składowej proporcjonalnej jako wzmocnienie krytyczne *Ku*, za pomocą których wyznacza się docelowe wartości nastaw według poniższej tabeli:

*Tabela 1: Wartości nastaw według metody Zieglera-Nicholsa*

|  |  |  |
| --- | --- | --- |
| **P** | **I** | **D** |
| 0.6*Ku* | *Pu*/2 | *Pu*/8 |

* + 1. Metoda Cohena-Coona

[PLACEHOLDER]

* 1. Przykładowe ćwiczenia

[PLACEHOLDER]

[PLACEHOLDER – W OPCJI ĆWICZEŃ ZASUGEROWAĆ DODAWANIE RADIATORÓW DO CZUJNIKA W CELU ZMIANY JEGO BEZWŁADNOŚCI]