Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Informatyka Czasu Rzeczywistego

Projekt semestralny

„Klimatyzator”

Autorzy:

Miłosz Błachowiak

Rafał Guzek

Kraków, 2023

Wstęp

Celem projektu było utworzenie układu sterującego klimatyzatorem opartego o platformę Arduino, który zarówno pod względem możliwości jak i udostępnianych funkcji, będzie w jak największym stopniu zbliżony do współcześnie projektowanych urządzeń. Ze względu na ograniczenia sprzętowe, całość projektu została zrealizowana w symulatorze systemów wbudowanych *Wokwi*.

Specyfikacja problemu

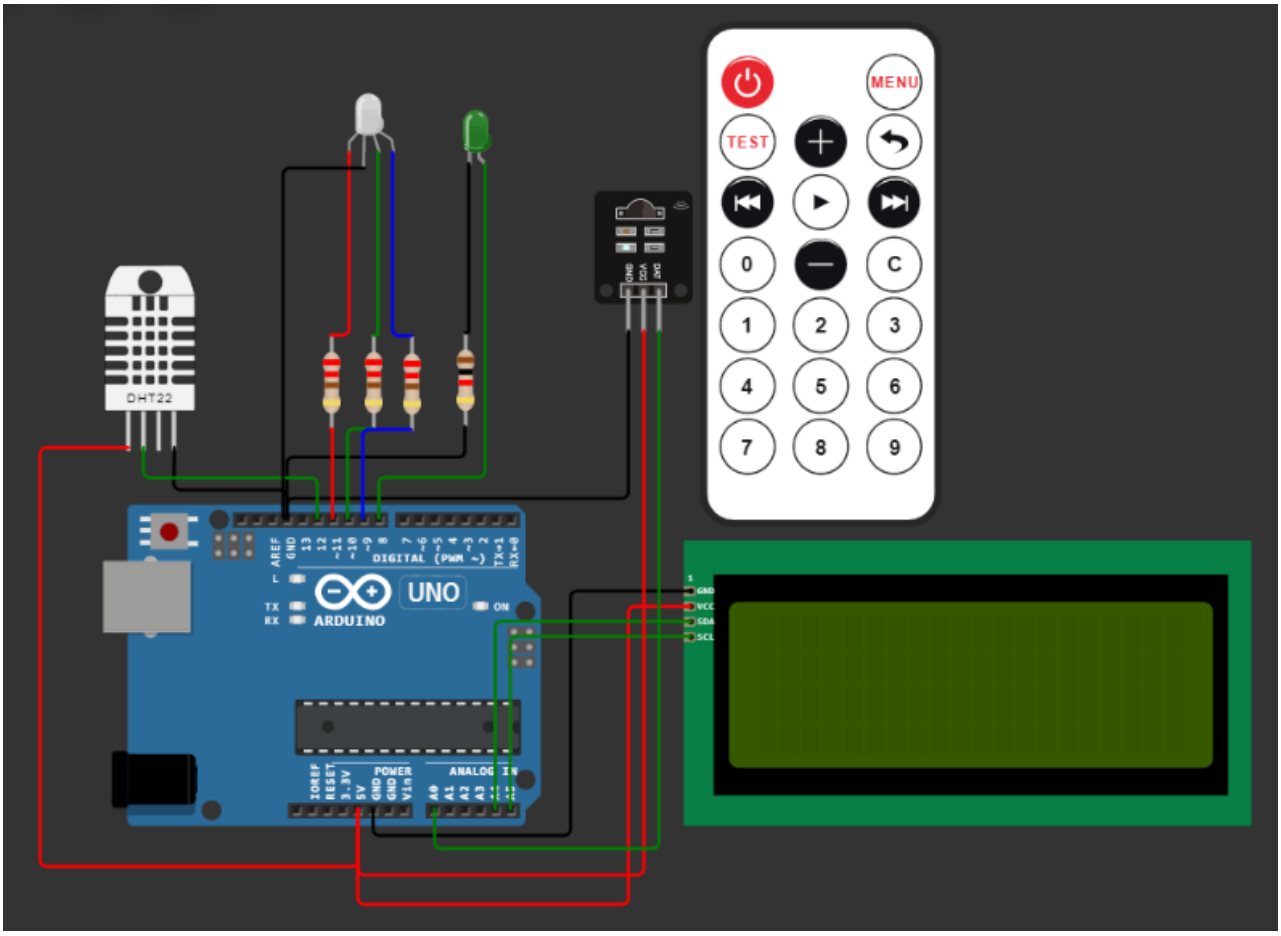
Projekt obejmuje implementację sterownika do klimatyzatora wykonanego w oparciu o mikrokontroler Arduino Uno. Sterownik daje możliwość zdalnej konfiguracji urządzenia przy pomocy pilota. Klimatyzator może działać w trzech trybach: nawiewu (wentylacja bez zmiany temperatury powietrza), chłodzenia i ogrzewania. Dla każdego z tych trybów możliwe jest ustawienie mocy (intensywności) wyrażonej w procentach. Regulacja temperatury w trybach chłodzenia i ogrzewania może być realizowana na dwa sposoby: manualnie - poprzez ręczne włączanie i wyłączanie urządzenia pilotem oraz manipulowanie wartością intensywności lub automatycznie. W przypadku regulacji automatycznej wykorzystywana jest implementacja regulatora dwupołożeniowego z histerezą. W zależności od wybranego trybu, chłodzenie lub grzanie wyłączane jest w sposób automatyczny, gdy temperatura zarejestrowana przez czujnik osiągnie wartość zadaną (z uwzględnieniem wartości histerezy). Oprócz tego urządzenie włącza się ponownie, gdy temperatura odpowiednio wzrośnie lub spadnie i uzyska wartość wykraczającą poza obszar histerezy. Użytkownik jest stale informowany o aktualnych parametrach pracy urządzenia za pomocą wyświetlacza LCD.

Budowa układu

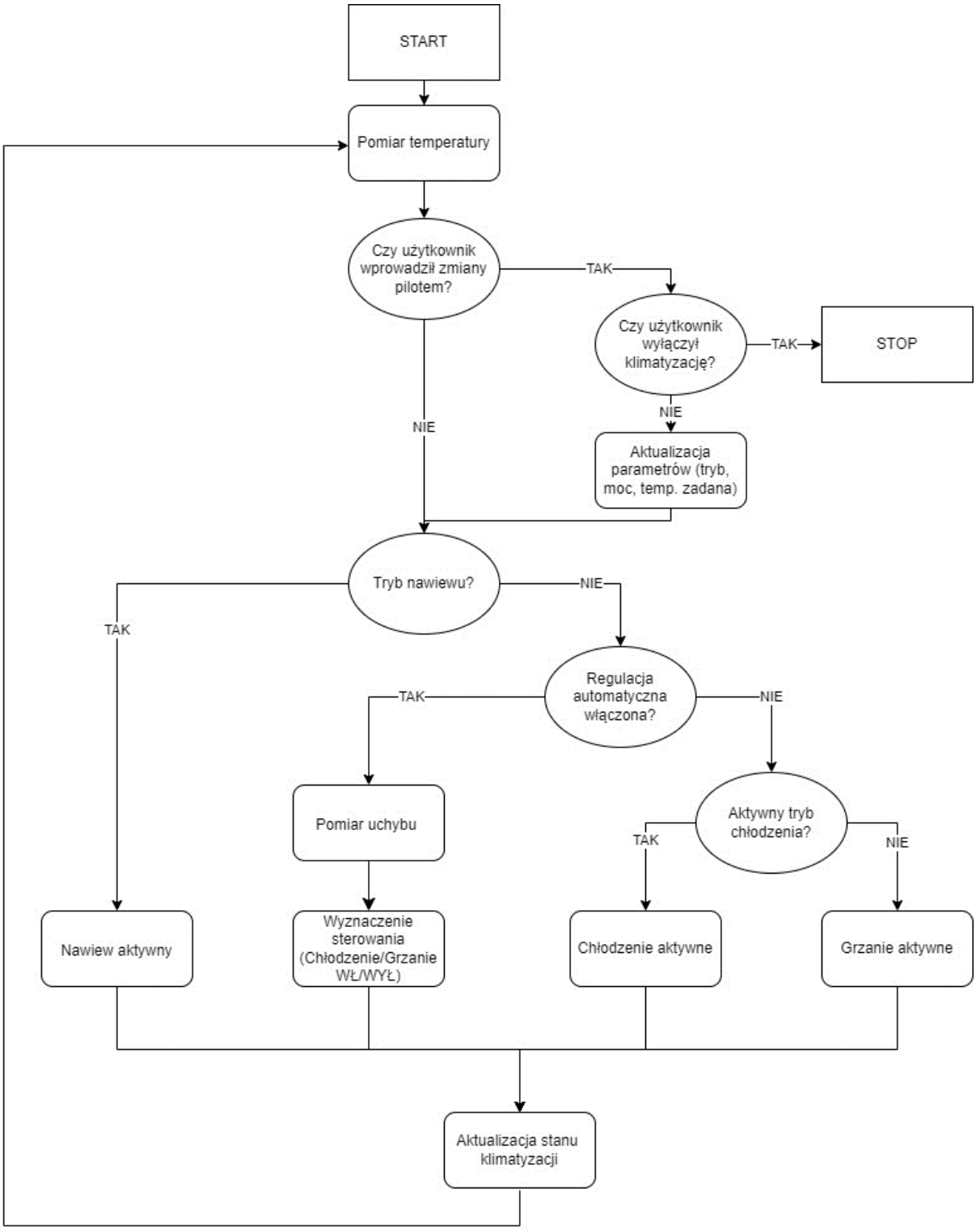
Najważniejszym elementem zbudowanego układu jest mikrokontroler Arduino Uno, z którego pomocą realizowana jest cała logika systemu sterującego klimatyzatorem. Układ wyposażono w czujnik temperatury i wilgotności DHT22, który stosowany jest do pomiaru temperatury otoczenia. Istotnym elementem jest również odbiornik podczerwieni, który wprowadza możliwość wykorzystania pilota do zdalnego sterowania klimatyzacją. Do układu dołączono również wyświetlacz LCD dający możliwość informowania użytkownika o aktualnych parametrach układu, takich jak tryb działania klimatyzacji, temperatura zadana i zmierzona oraz moc z jaką działa urządzenie. W celu zasymulowania samej klimatyzacji wykorzystano dwie diody LED. Zielona dioda informuje nas o tym, czy urządzenie jest włączone. Natomiast dioda RGB wskazuje aktualny tryb w jakim działa klimatyzacja – kolor biały oznacza tryb nawiewu, niebieski to tryb chłodzenia, natomiast czerwony to ogrzewanie.

Schemat połączeń

Poniżej przedstawiono ogólny schemat układu oraz połączenia pomiędzy poszczególnymi elementami składowymi systemu.



Schemat blokowy algorytmu



Sposób realizacji

Oprogramowanie realizujące funkcję sterownika do klimatyzacji zostało napisane w języku C. W implementacji projektu wykorzystano wymienione w poniższym fragmencie kodu biblioteki, które zastosowano do obsługi urządzeń wchodzących w skład systemu.

```
#include "DHT.h"
#include <Wire.h>
#include <IRremote.h>
#include <LiquidCrystal_I2C.h>
```

Niezbędnym krokiem było zdefiniowanie pinów, po których odbywać się będzie komunikacja z urządzeniami podłączonymi do mikrokontrolera Arduino Uno. Komunikację zainicjalizowano w funkcji *setup*.

```
#define PIN_RECEIVER A0
#define DHTPIN 12
#define DHTTYPE DHT22
#define STATUS_LED_PIN 8

#define RGB_RED 11
#define RGB_GREEN 10
#define RGB_BLUE 9

DHT dht(DHTPIN, DHTTYPE);
IRrecv receiver(PIN_RECEIVER);
LiquidCrystal_I2C LCD(0x27, 20, 4);

[…]

void setup() {
  Serial.begin(115200);
  Serial.println("Started!");

  dht.begin();
  LCD.init();

  pinMode(RGB_RED, OUTPUT);
  pinMode(RGB_GREEN, OUTPUT);
  pinMode(RGB_BLUE, OUTPUT);

  receiver.enableIRIn();

  […]

}
```

Za kontrolę przebiegu całego algorytmu odpowiada wykonywana w pętli funkcja *loop,* wewnątrz której wywoływane są kolejno funkcje odpowiadające za obsługę poszczególnych elementów całego systemu.

```
void loop() {

  DHT_temperature_C = dht.readTemperature();

  AirConditionerStatus prev_status = airConditionerParams.status;
  AirConditionerMode prev_mode = airConditionerParams.mode;

  // Checks received an IR signal
  if (receiver.decode()) {
    handle_air_conditioning(&receiver, &airConditionerParams);
    receiver.resume();  // Receive the next value
  }

  if (airConditionerParams.regulation == AUTOMATIC) {
    calc_regulator_output(&airConditionerParams, DHT_temperature_C);
  }

  if (airConditionerParams.status != prev_status) {
    change_target_AC_status(airConditionerParams.status);
  }

  if (airConditionerParams.mode != prev_mode) {
    change_target_AC_mode(airConditionerParams.mode);
  }

  if (airConditionerParams.status == ON) {
    if (prev_status == OFF) {LCD.backlight();}
    handle_display(&airConditionerParams);
  }
  else {
    if (prev_status == ON) {
      LCD.clear();
      LCD.noBacklight();
    }
  }
  delay(DELAY_MS);
}
```

Na początku każdej pętli pobierana jest wartość temperatury rejestrowanej przez sensor DHT 22 poprzez wywołanie metody *readTemperature*. Następnie sprawdzane jest, czy użytkownik wprowadził zmiany za pomocą któregokolwiek z przycisków na pilocie. Jeśli takie zdarzenie miało miejsce, następuje aktualizacja parametrów urządzenia w funkcji *handle_air_conditioning*.

```
void handle_air_conditioning(IRrecv* receiver, AirConditionerParams*
airConditionerParams) {
  if (!receiver) {
    Serial.println("IR ERROR\n");
    return;
  }

  if (!airConditionerParams) {
    Serial.println("air-conditioner parameters ERROR\n");
    return;
  }
```

```
  if (receiver->decodedIRData.command == POWER) {
    handle_power(airConditionerParams);
    return;
  }

  if (airConditionerParams->status == OFF) {
    return;
  }

  switch (receiver->decodedIRData.command) {
    case TEST: // plus temp
      handle_temperature_turn_up(airConditionerParams);
      break;
    case num_0: // minus temp
      handle_temperature_turn_down(airConditionerParams);
      break;
    case PLUS: // HEAT
      handle_heating(airConditionerParams);
      break;
    case PLAY: // WIND
      handle_wind(airConditionerParams);
      break;
    case MINUS: // COOL
      handle_cooling(airConditionerParams);
      break;
    case BACK: // plus power
      handle_ventilationPower_turn_up(airConditionerParams);
      break;
    case key_C: // minus power
      handle_ventilationPower_turn_down(airConditionerParams);
      break;
    case MENU: // regulation mode
      handle_regulation_mode(airConditionerParams);
      break;
    default:
      Serial.println("Not Implemented\n");
      break;
  }
}
```

Gdy ustawiony jest tryb automatycznej regulacji, wywoływana jest funkcja *calc_regulator_output,* która wyznacza sterowanie w oparciu o implementację regulatora dwupołożeniowego z histerezą. Sterowanie może przyjmować jedynie wartości 0 (zmierzona temperatura otoczenia osiągnęła wartość zadaną) lub 1 (potrzebne jest schłodzenie/ogrzanie powietrza w celu osiągnięcia temperatury zadanej). W pierwszym przypadku klimatyzator jest wyłączany – moc urządzenia ustawiana jest na 0%, natomiast w drugim działa on z mocą ustawioną przez użytkownika.

```
void calc_regulator_output(AirConditionerParams* ac_params, float
sensor_temperature){

    float error = ac_params->set_temperature - sensor_temperature;

    if (ac_params->mode == COOL){
      if (error > hysteresis / 2 && output == 1) {
        output = 0;
      }

      if (error < -hysteresis / 2 && output == 0) {
        output = 1;
      }
    }
    else if (ac_params->mode == HEAT){
      if (error > hysteresis / 2 && output == 0) {
        output = 1;
      }
      if (error < -hysteresis / 2 && output == 1) {
        output = 0;
      }
    }
    else {
      output = 1;
    }
}
```

Funkcje *change_target_AC_status* i *change_target_AC_mode* aktualizują obecny stan i tryb działania urządzenia w oparciu o zmiany wprowadzone przez użytkownika oraz wyznaczone sterowanie. Za wizualizację aktualnego stanu i trybu klimatyzatora odpowiedzialne są dwie diody LED.

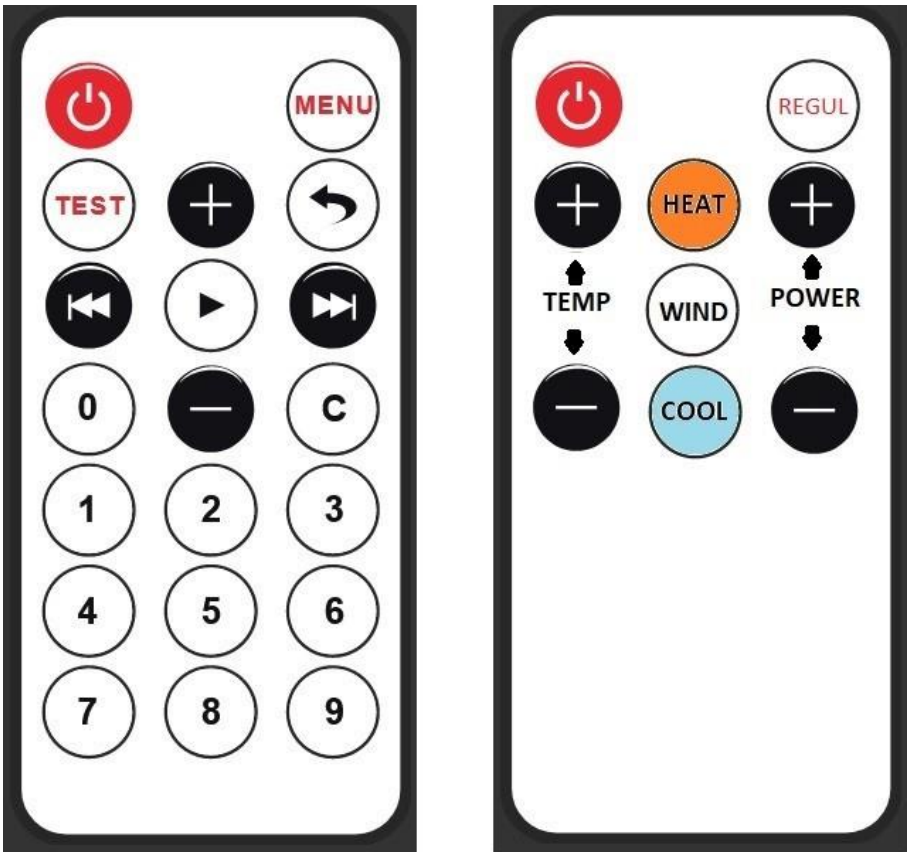
Końcowym etapem w każdym przebiegu pętli algorytmu jest obsługa wyświetlacza LCD, która realizowana jest przez funkcję *handle_display.*

```
void handle_display(AirConditionerParams* ac_params) {

  if (!ac_params) {
    Serial.println("AirConditioner parameters ERROR!");
  }

  LCD.setCursor(0, 0);
  if (ac_params->regulation == AUTOMATIC) {
    LCD.print("POWER [%]: "); LCD.print(ac_params->set_ventilation_power *
output); LCD.print(" ");
  }
  else {
    LCD.print("POWER [%]: "); LCD.print(ac_params->set_ventilation_power);
LCD.print(" ");
  }

  LCD.setCursor(0, 1);
  LCD.print("REGULATION: "); LCD.print(acRegulation2Str(ac_params-
>regulation)); LCD.print("  ");
```

```
  if (isnan(DHT_temperature_C) || isnan(DHT_humidity)) {
    Serial.println("Error while reading temperature and humidity
measurements!");
  }
  else
  {
    LCD.setCursor(0, 2);
    LCD.print("CURR TEMP [C]: "); LCD.print(DHT_temperature_C);

    if (ac_params->regulation == AUTOMATIC) {
      LCD.setCursor(0, 3);
      LCD.print("SET TEMP [C]: "); LCD.print(ac_params->set_temperature);
    }
    else {
      LCD.setCursor(0, 3);
      LCD.print("                    ");
    }
  }
}
```

Konfiguracji przycisków na pilocie

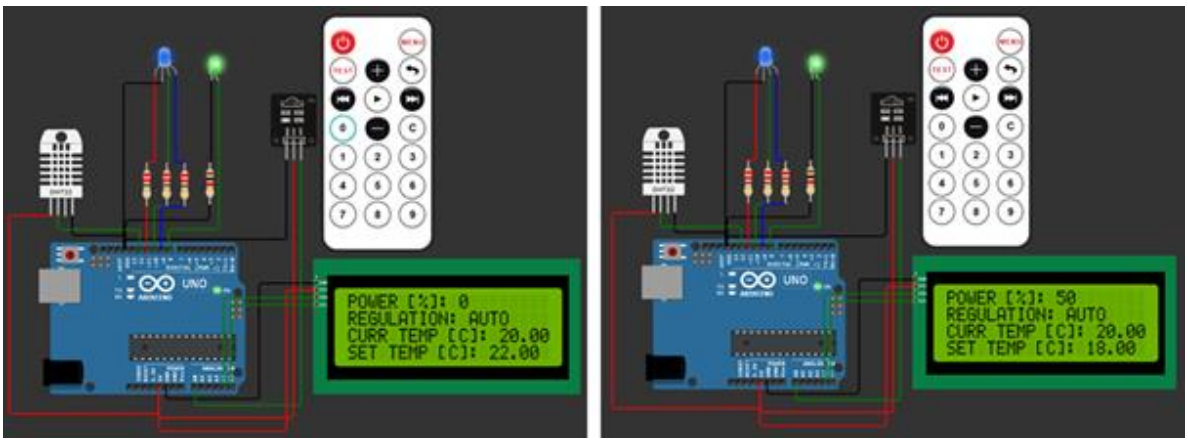
Poniżej przedstawiono porównanie pomiędzy oznaczeniami przycisków na pilocie w *Wokwi* (po lewej stronie), a ich rzeczywistymi funkcjami (po prawej).



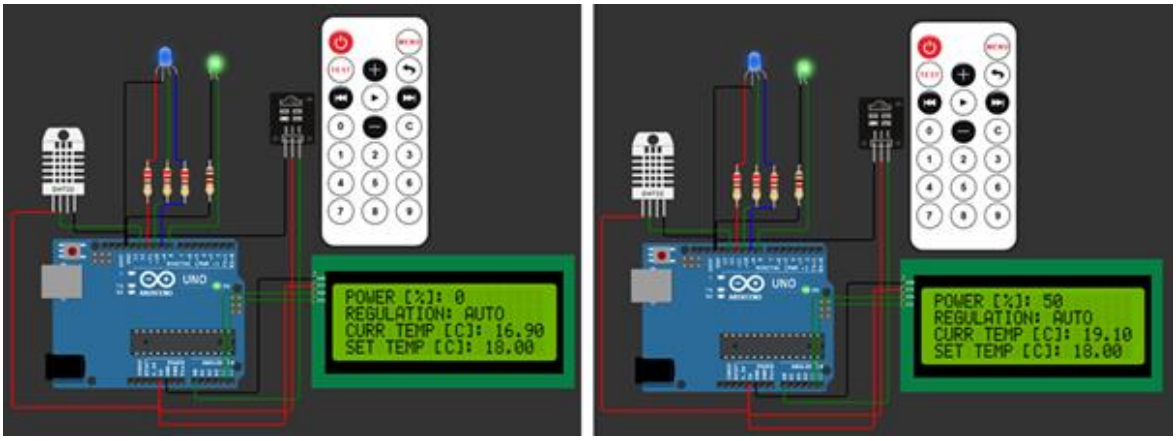
Testy oprogramowania

Tryb chłodzenia (COOL):

Testowanie trybu chłodzenia rozpoczęte zostało od ustawienia temperatury zadanej na wyższą od temperatury otoczenia, tak aby zweryfikować, czy klimatyzacja w tym przypadku zostanie odłączony (moc klimatyzacji zostanie zmieniona na 0%). Następnie temperatura zadana została znacznie zmniejszona, aby sprawdzić czy klimatyzacja zostanie załączona (moc nie będzie równa 0%).

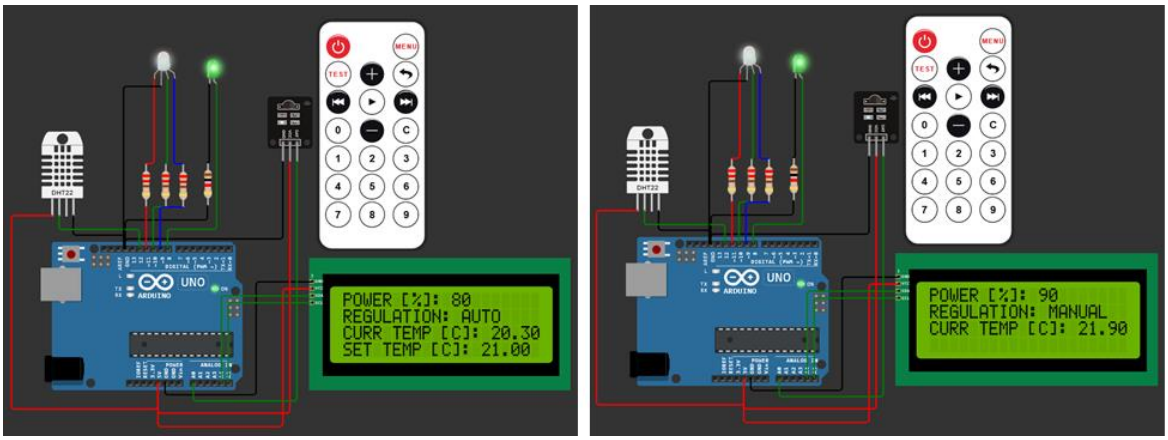


Ostatnie eksperymenty związane z omawianym trybem dotyczyły sprawdzenia poprawności działania regulatora dwupołożeniowego z histerezą. W tym celu temperatura otoczenia była ręcznie zmieniana, poprzez manipulowanie odczytem z czujnika DHT22 do momentu osiągnięcia wybranej granicy histerezy.



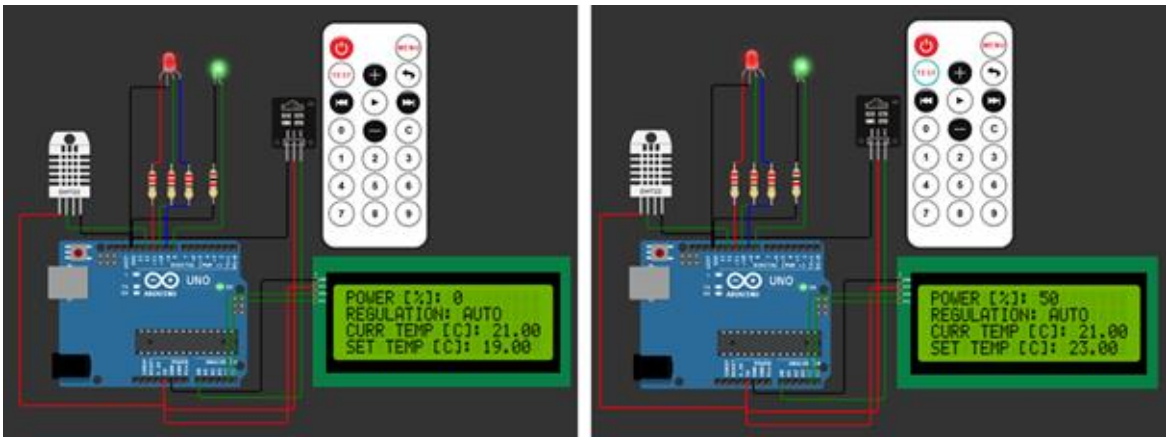
Tryb Nawiewu (WIND):

Ze względu na to, iż tryb nawiewu nie wpływa bezpośrednio na zmianę temperatury otoczenia, przeprowadzone testy opierały się w głównej mierze na manipulowaniu mocą nawiewu.

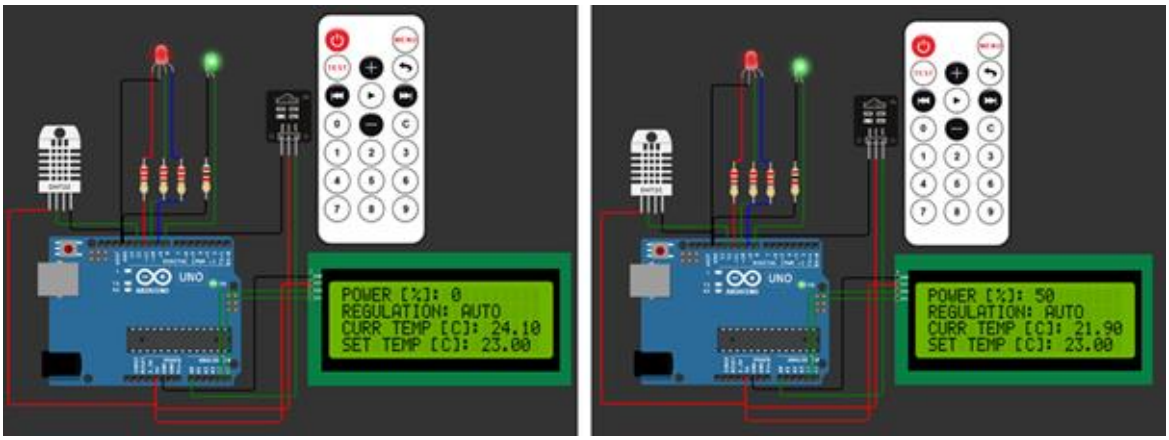


Tryb grzania (HEAT):

Testowanie trybu grzania zaczęte zostało od ustawienia temperatury zadanej na niższą niż temperatura otoczenia, tak aby sprawdzić, czy funkcja grzania zostanie w tej sytuacji wyłączona (moc klimatyzacji zostanie zmieniona na 0%). Później temperatura zadana została zwiększona, aby zweryfikować poprawność załączania się badanego trybu (moc nie będzie równa 0%).

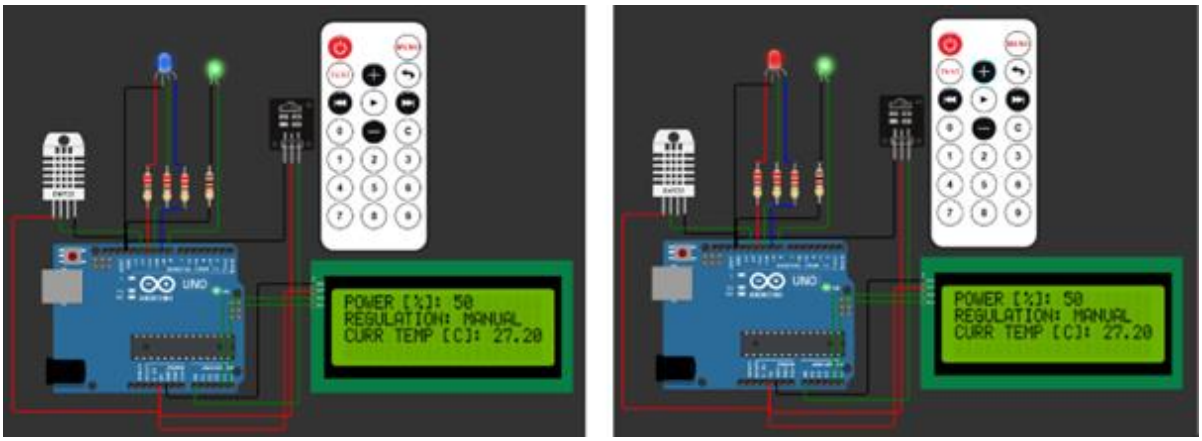


Końcowe testy trybu ogrzewania dotyczyły sprawdzenia poprawności działania regulatora dwupołożeniowego z histerezą. W tym celu zmianie poddawana została jedynie wartość temperatury otoczenia do momentu osiągnięcia zarówno pierwszej jak i drugiej granicy wyznaczanej przez wprowadzoną histerezę.



Sterowanie ręczne (MANUAL):

Zważywszy na to, że w trybie ręcznym moc nawiewu nie jest kontrolowana przez regulator, tylko przez użytkownika, to temperatura zadana nie ma wpływu na działanie klimatyzatora, dlatego też nie jest wyświetlana na ekranie LCD. Biorąc to pod uwagę testy w tym trybie sprowadzały się jedynie do zmiany trybu działania klimatyzacji (grzanie/nawiew/chłodzenie), a także odpowiednim manipulowaniu wartością mocy nawiewu.



Propozycje rozwoju

Rozpatrując zaprojektowany program można zauważyć, iż jest on jedynie prostym prototypem systemu regulującego temperaturę otoczenia i nie udostępnia wszystkich nowoczesnych funkcji, pojawiających się w dzisiejszych klimatyzatorach. Do jednych z takich funkcji należy tryb osuszania i oczyszczania powietrza, który pozwala na usunięcie nadmiaru wilgotności z powietrza, a także zapobiega rozprzestrzenianiu się drobnoustrojów, zarodników pleśni i grzybów oraz roztoczy kurzu domowego. Drugim udogodnieniem udostępnianym przez klimatyzatory stanowi możliwość zaprogramowania urządzenia w taki sposób, aby włączało się ono i wyłączało w konkretnych godzinach ustalonych przez użytkownika. Oprócz tego dzisiejsze urządzenia pozwalają również na ręczne ustawianie kierunku nawiewu, poprzez kontrolowanie pozycji łopatek klimatyzatora. Pomimo, że przedstawione funkcje mogą wydawać się skomplikowane, to ich odzwierciedlenie w wykorzystywanym symulatorze jest możliwe i może stanowić propozycję rozwoju omawianego projektu.

Podsumowanie

W ramach projektu, utworzony został układ działający zarówno w trybie ręcznym, jak i automatycznym, pozwalający na regulację temperatury pomieszczenia oraz kontrolowanie mocą nawiewu. Analizując zaprojektowany program można zauważyć, iż wspiera on jedynie główne oraz najważniejsze funkcje dzisiejszych systemów klimatyzacji, lecz nie udostępnia żadnych dodatkowych funkcji, które w niektórych przypadkach mogą stanowić pewną kartę przetargową dla potencjalnego konsumenta.

Realizacja przedstawionego projektu, pozwoliła nam na rozwinięcie swoich umiejętności dotyczących programowania w języku “C”, a także dała nam okazję do zapoznania się z symulatorem sprzętu elektronicznego jakim jest środowisko *Wokwi*.