

**ТЕХНОЛОГИЧНО УЧИЛИЩЕ „ЕЛЕКТРОННИ СИСТЕМИ“  
Към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**КУРСОВА РАБОТА  
ПО УВОД ВЪВ ВГРАДЕНИ МИКРОКОМПЮТЪРНИ СИСТЕМИ**

**Тема: Система за дистанционно контролиране на компютър или  
домашен сървър**

**Курсист:**  
Калоян Дойчинов

**Оценител:**  
Енчо Шахънов

**гр. София  
2022**



Технологично училище  
„Електронни системи“ към  
Технически университет – гр. София

Дата на заданието: 19.05.2022г.  
Дата на предаване: 19.05.2022г.

Утвърждавам:.....  
/г-н Енчо Шахънов/

# ЗАДАНИЕ

## за курсова работа

на ученика Калоян Стефанов Дойчинов от 10а клас

**1. Тема:** Дистанционно управление на компютър

**2. Изисквания**

- 2.1 Разработване на работещ хардуерен проект, включващ използването на Ардуино платка
- 2.2 Разработване на система, която управлява работното състояние на компютър и отчита данни като температура и влага
- 2.3 Визуализиране на данни като измерената температура и влага в компютърната кутия

Курсист:.....  
/Калоян Дойчинов/

Оценител:.....  
/г-н Енчо Шахънов/

# Съдържание

<b>УВОД .....</b>	- 4 -
<b>ПЪРВА ГЛАВА.....</b>	- 5 -
<b>ХАРДУЕРНИ КОМПОНЕНТИ, ИЗПОЛЗВАНИ В РЕАЛИЗАЦИЯТА НА ПРОЕКТА ..</b>	- 5 -
<b>1.1. Избиране на Ардуино платка.....</b>	- 5 -
<b>1.1.1. Ардуино Уно ревизия 3.....</b>	- 5 -
<b>1.1.2. Ардуино Нано .....</b>	- 6 -
<b>1.2. Избиране на компонент, който ще включва и изключва компютър .....</b>	- 6 -
<b>1.3. Избиране на сензор за температура и влага .....</b>	- 7 -
<b>1.3.1. DHT11 и DHT22.....</b>	- 7 -
<b>1.4. Избиране на компонент за получаване и изпращане на данни към клиента .....</b>	- 7 -
<b>1.4.1. Избиране на протокол за връзка .....</b>	- 7 -
<b>1.4.2. Избиране на Wi-Fi модул.....</b>	- 8 -
<b>1.4.3. ESP-01 (ESP8266-01) .....</b>	- 9 -
<b>1.5. Избиране на захранване.....</b>	- 10 -
<b>ВТОРА ГЛАВА .....</b>	- 11 -
<b>БЛОК СХЕМА НА ПРОЕКТА .....</b>	- 11 -
<b>2.1 Блок схеми на работа на проекта.....</b>	- 11 -
<b>ТРЕТА ГЛАВА.....</b>	- 13 -
<b>РЕАЛИЗАЦИЯ И РАЗРАБОТВАНЕ НА ПРОЕКТА .....</b>	- 13 -
<b>3.1 Описание на схема на свързване.....</b>	- 13 -
<b>3.1.1 Свързване на бредбординг и захранващ модул .....</b>	- 13 -
<b>3.1.2 Свързване на DHT11 .....</b>	- 13 -
<b>3.1.3 Свързване на реле.....</b>	- 13 -
<b>3.1.4 Свързване на ESP-01 .....</b>	- 14 -
<b>3.1.5 Описание на софтуерната част на проекта.....</b>	- 14 -
<b>3.2 Файлова структура на проекта .....</b>	- 14 -
<b>3.3 Софтуерът за Ардуино платката.....</b>	- 14 -
<b>3.4 Софтуерът за сървъра .....</b>	- 21 -
<b>3.4.1 Уеб клиент.....</b>	- 21 -
<b>3.4.2 Гласов клиент.....</b>	- 22 -
<b>3.5 Уеб клиента и гласовите клиенти.....</b>	- 23 -
<b>ЧЕТВЪРТА ГЛАВА .....</b>	- 25 -
<b>СЪЗДАВАНЕ НА РАБОТОСПОСОБЕН МОДЕЛ НА ПРОЕКТА .....</b>	- 25 -
<b>ЗАКЛЮЧЕНИЕ .....</b>	- 27 -
<b>ИЗПОЛЗВАНА ЛИТЕРАТУРА .....</b>	- 28 -

## **УВОД**

С напредването на технологиите и вграждането им в нашия свят все повече и повече неща трябва да се контролират от разстояние – компютрите на работниците в различни фирми, които искат да позволят отдалечен достъп на своите служители до хората, които трябва да достъпят даден важен документ, презентация, файл, който са забравили на настолния си компютър и биха могли да се свържат безжично или на хората с домашни или отдалечени DIY<sup>i</sup> сървъри. Тези хора разчитат на отдалечената връзка с подобни компютри, което ги прави уязвими, когато забравят да ги включват, токът спре за няколко минути и след да трябва някой да ги включва ръчно или искат да автоматизират включването на дадените компютри. Тази курсова работа има за цел да реши този проблем.

# ПЪРВА ГЛАВА

## ХАРДУЕРНИ КОМПОНЕНТИ, ИЗПОЛЗВАНИ В РЕАЛИЗАЦИЯТА НА ПРОЕКТА<sup>ii</sup>

### 1.1. Избиране на Ардуино платка

#### 1.1.1. Ардуино Уно ревизия 3<sup>iii</sup>

Ардуино Уно е най–достъпната, удобна и лесна за програмиране и употреба платка както за начинаещи, така и за напреднали, правейки това най–разпространената Ардуино платка. Поради тази причина има изключително много информация за това как работи и как трябва да се програмира в интернет. Освен това заедно с платката е предоставена и лесна и интерактивна среда за програмиране – Ардуино IDE, осигуряваща всичко необходимо откъм допълнителни библиотеки и софтуер свързани с програмирането на платката. Тези фактори я правят добър избор за начинаещи – от малки до големи.

Поради горепосочените причини в този проект се използва платката Ардуино Уно рев. 3 (*фиг. 1*), която е най–новата разновидност на Ардуино Уно. Ардуино Уно рев. 3 е базирана на микроконтролера ATmega328P, който е 8-битов, и има 16MHz резонатор – от резонаторът зависи бързодействието на платката. Микроконтролерът не е запоен за платката и това позволява да бъде сменен много лесно в случай, че се случи нещо или Ардуино платката трябва да бъде ползвана като USB Bus устройство за програмиране на други платки като ESP-01.

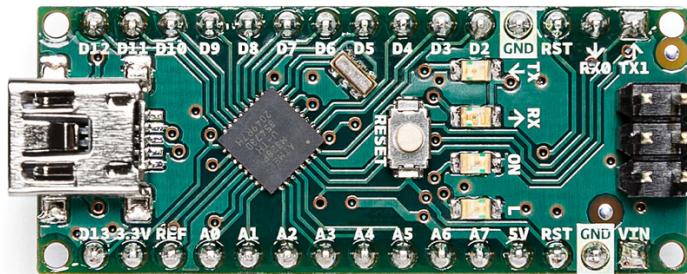
Тя има общо 14 цифрови пина, които могат да служат както за входове, така и за изходи. Освен това има и 6 аналогови, които обаче не са използвани в този проект.



*Фиг. 1 – Ардуино Uno Rev. 3*

### 1.1.2. Ардуино Нано<sup>iv</sup>

Друга отлична платка за начинаещи е Ардуино Нано (фиг. 2). Главната разлика между тази платка и Ардуино Уно рев. 3 е, че заема сравнително малка площ (45x18mm срещу 68.6x53.3mm на Уно). Това би направило проекта да може да се поставя в по-малки и ограничени компютърни кутии.



Фиг. 2 – Ардуино Нано

### 1.2. Избиране на компонент, който ще включва и изключва компютър

Релето е електрически задвижван ключ. То използва електромагнит (намотка), за да управлява вътрешния си механичен превключващ механизъм (контакти). Когато релейният контакт е отворен, това ще включи захранването за верига, когато бобината е активирана.



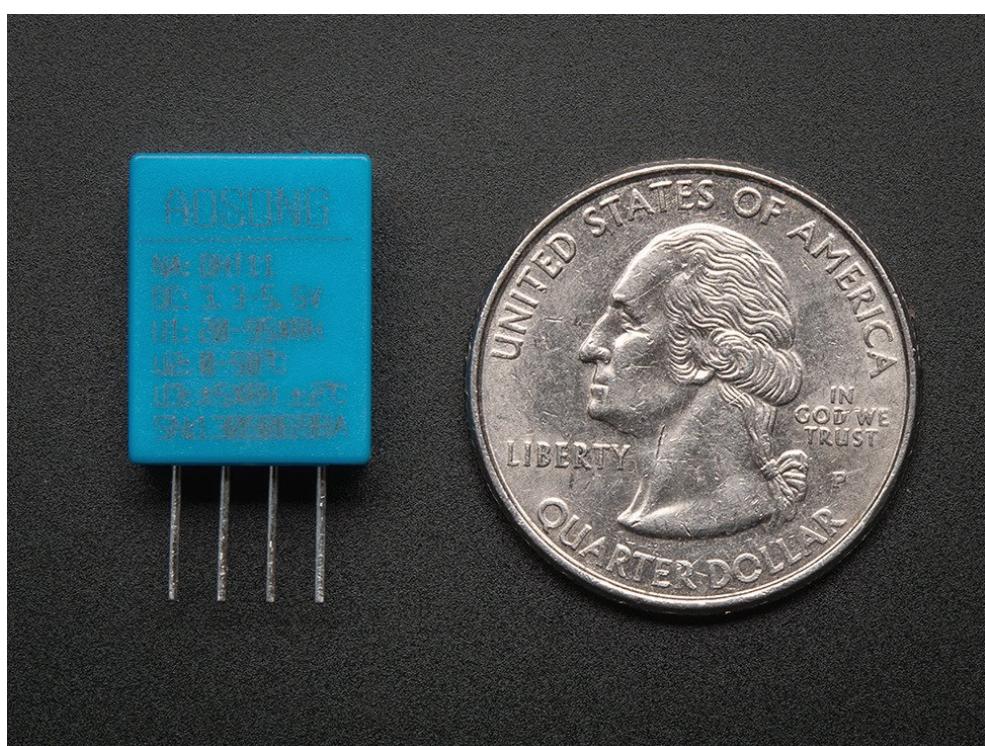
Фиг. 3 - Реле

## **1.3. Избиране на сензор за температура и влага**

### **1.3.1. DHT11 и DHT22<sup>v</sup>**

DHT11 е прост и евтин цифров сензор за измерване на температура и влажност. Той използва капацитивен сензор за влажност и термистор за измерване на околнния въздух, като изпраща сигнал върху цифровите пинове за данни, съкращавайки нуждата от използване на АЦП. Единственият недостатък на този сензор е, че може да се получават нови данни от него всеки 2 секунди.

В сравнение с DHT22, DHT11 е с по-малка прецизност, с по-малка точност и работи в по-малък диапазон за температура/влажност – от 0° до 50°C и от 20% до 90% влага. В същото време е сравнително по-малък и с по-ниска цена.



*Фиг. 4 – DHT11 и американски четвърт долар*

## **1.4. Избиране на компонент за получаване и изпращане на данни към клиента**

### **1.4.1. Избиране на протокол за връзка**

Както Bluetooth, така и Wi-Fi (фиг. 6) се използват за осигуряване на безжична комуникация чрез радиосигнали. Основната разлика между Bluetooth и Wi-Fi е, че Bluetooth се използва за свързване на устройства с

малък обсег за споделяне на информация, докато Wi-Fi се използва за предоставяне на високоскоростен достъп до цялата мрежа - интернет. Различията между двата стандарта са показани на таблицата от *фиг. 5*:

<b>Bluetooth</b>	<b>Wi-Fi</b>
Bluetooth консумира малко енергия.	Wi-Fi консумира висока мощност.
Сигурността на Bluetooth е по-ниска в сравнение с Wi-Fi.	Wi-Fi осигурява по-добра сигурност от Bluetooth.
Bluetooth поддържа ограничен брой потребители	Wi-Fi поддържа голям брой потребители.
Обхватът на радиосигнала на Bluetooth е десет метра.	Wi-Fi обхватът може да достигне десетки метри, зависейки от честотната лента.
Bluetooth изисква ниска честотна лента.	Wi-Fi изисква висока честотна лента.
Bluetooth е т.нар. PAN – Personal Area Network – използва се в подобни IoT решения, но има лимит от 10м за контролиране	Wi-Fi може да се свърже към т.нар. WAN – Wide Area Network – което означава, че се свързва към интернет, правейки го така, че да може да се контролира независимо къде потребителят се намира.

*Фиг. 5 – таблица за сравнение между Wi-Fi и Bluetooth*

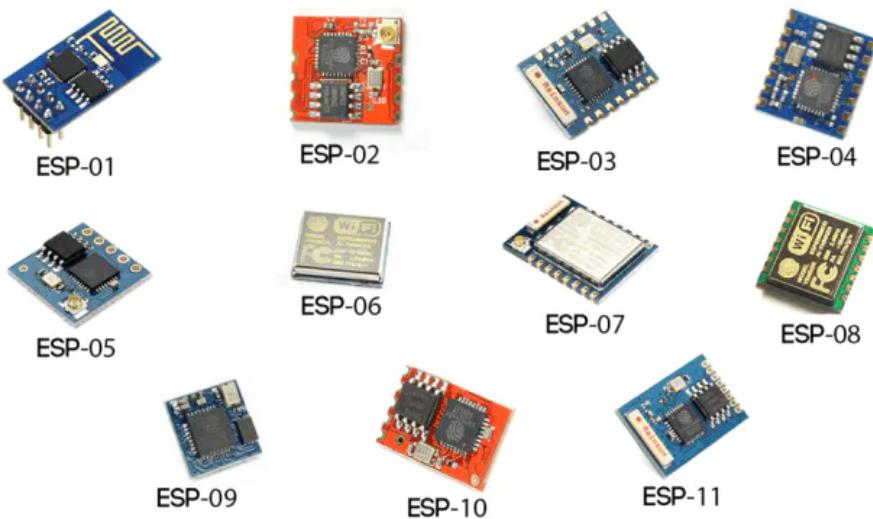


*Фиг. 6 – Bluetooth и Wi-Fi*

#### 1.4.2. Избиране на Wi-Fi модул

Има много ESP Wi-Fi модели, затова е важно да се избере най-подходящият. За този проект всеки един от Wi-Fi модулите от семейството

на ESP (фиг. 7) би бил подходящ, затова е избран най-лесният и най-достъпен модул – ESP-01.

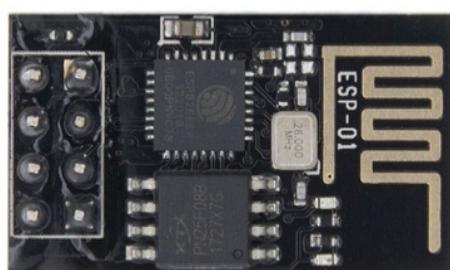


Фиг. 7 – ESP семейството

#### 1.4.3. ESP-01 (ESP8266-01)<sup>vi</sup>

ESP8266-01 (фиг. 8) е най-популярният Wi-Fi модул, поради ниската си цена, в резултат на което има и огромна онлайн общност, в която може да се намери отговор на почти всеки въпрос или проблем, с който може човек да се сблъска.

ESP8266-01 или ESP-01 е Wi-Fi модул, който позволява на микроконтролерите като Ардуино Уно и др. достъп до Wi-Fi мрежа и достъп до интернет. Този модул е самостоятелен SoC<sup>vii</sup>, който не се нуждае непременно от микроконтролер за манипулиране на входове и изходи, защото действа като малък компютър. Освен това е много лесен за използване с Ардуино платките, тъй като е сериен Wi-Fi модул – може да комуникира с Ардуино платките през серийната комуникация.



Фиг. 8 – ESP8266-01

## 1.5. Избиране на захранване

След свързване на ESP-01 се установява как Ардуино Уно не може да подава достатъчно захранване, чрез +3.3V си захранване. Това може да се провери с мултициет, който измерва спад в +3.3V напрежение до около +2.2V ~ +2.4V. Заради това към проекта се добавя външно захранване. Едно от най-удобните захранвания е бредборд захранването, което може чрез шунтиране на дадени негови пинове да се надстрои да подава или +3.3V, или 5V, зависейки за коя захранваща линия се надстрои. Захранващия блок за бредборд, използван в този проект, е КИТ K2227<sup>viii</sup>, което може да бъде захранено или с Mini USB-B, или чрез букса (5.5мм/2.1мм), като чрез вградени светодиоди може да се види дали получава захранване.



Фиг. 9 – бредборд захранване

Този проект се захранва от 2 USB порта:

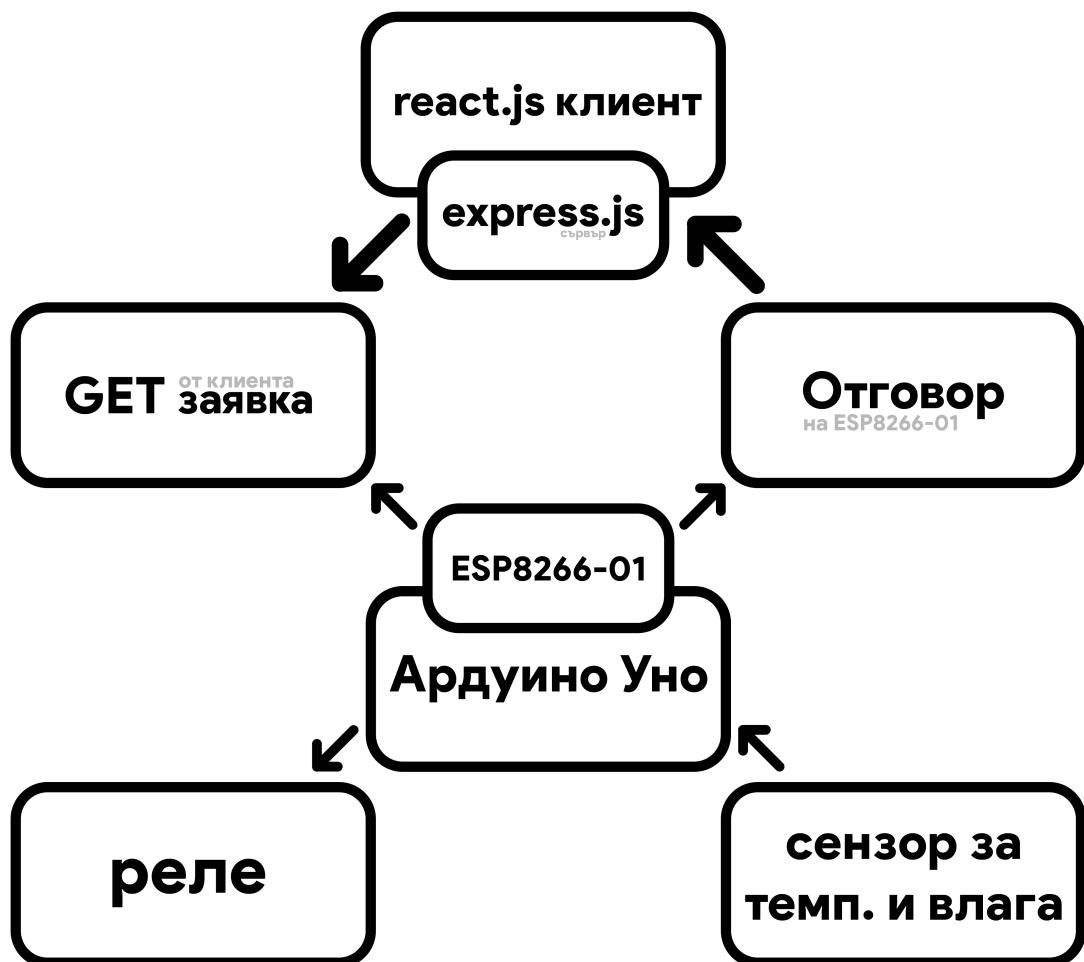
- USB-B на Ардуино платката
- Mini USB-B на бредборд захранването

## ВТОРА ГЛАВА

### БЛОК СХЕМА НА ПРОЕКТА

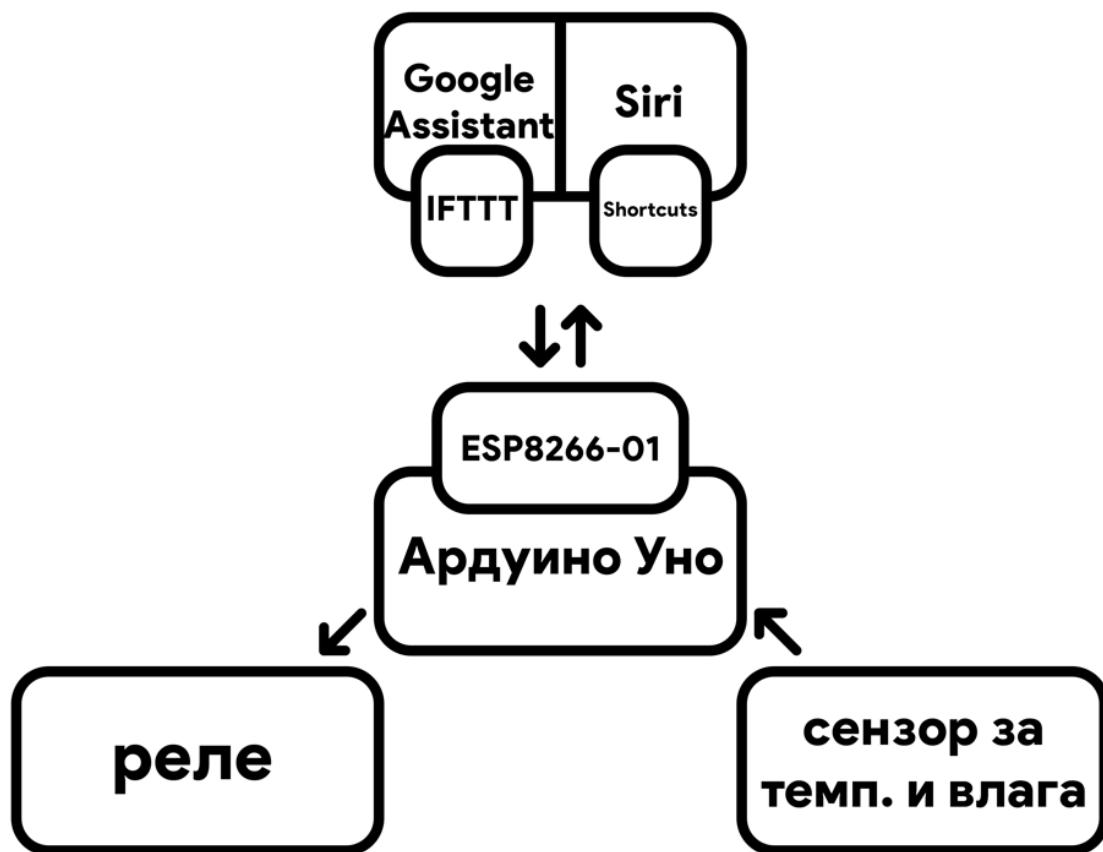
#### 2.1 Блок схеми на работа на проекта

Потребителите, които използват уеб приложението (фиг. 10), могат да изберат между 3 команди – да включат / изключват дадена машина (1), да получат данни за средата – температура и влага (2) или да нулират / рестартират дадена машина (3). При избиране на команда се изпраща GET заявка към Wi-Fi модула, който след това предава на Ардуино платката. При избор на включване/изключване или нулиране на компютъра, релето се използва като прекъсвач и шунтира връзката между двата пина на дънната платка съответно за 1. или 7 секунди. След това обратно се изпраща отговор за успешно изпълнение на командата. При избор на получаване на данни Ардуино платката комуникира с DHT11 сензорът за температура и влага и изпраща данните като отговор на заявката и я приключва успешно.



Фиг. 10 – блок схема на работа на уеб приложението с хардуера

Потребителите, които използват гласови асистенти (*фиг. 11*), имат същата достъпност както потребителите на уеб приложението. Обаче не се използва приложно-програмен интерфейс<sup>ix</sup> (express.js), а приложно-програмените интерфейси на IFTTT<sup>x</sup> за Google Assistant<sup>xi</sup> или Shortcuts<sup>xii</sup> за Siri<sup>xiii</sup> на Apple.



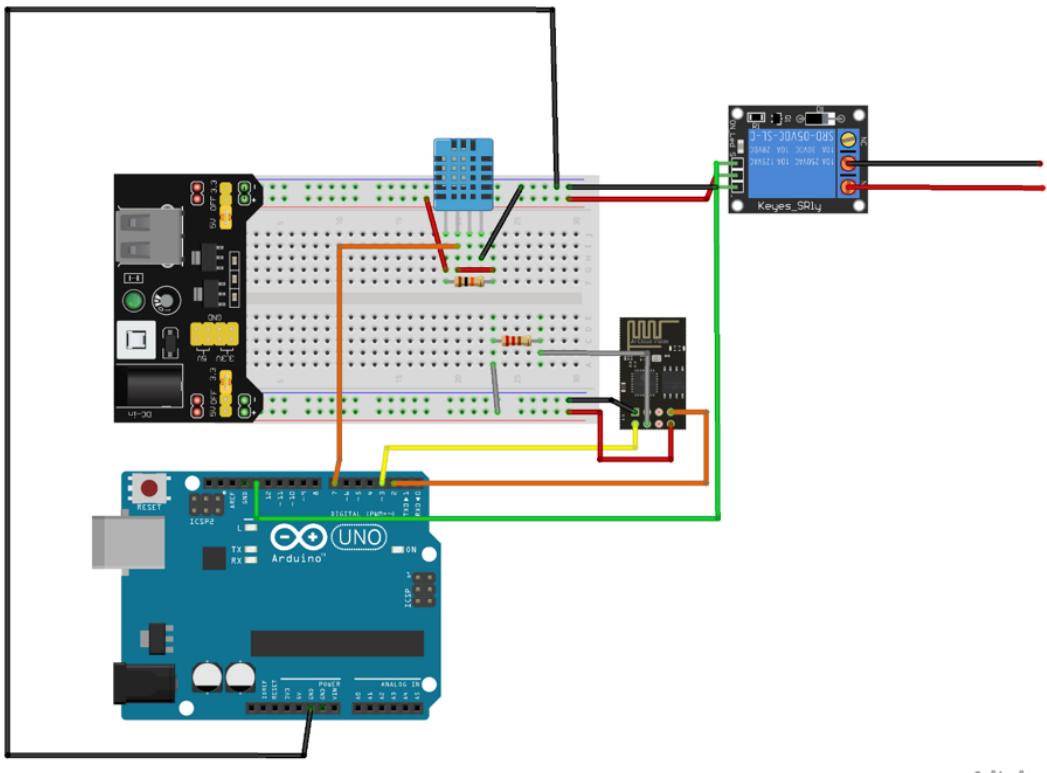
*Фиг. 11 – блок схема на работа с гласовите асистенти с хардуера*

## ТРЕТА ГЛАВА

# РЕАЛИЗАЦИЯ И РАЗРАБОТВАНЕ НА ПРОЕКТА

### **3.1 Описание на схема на свързване**

Схемата на свързване е показана на *фигура 12.*



Фиг. 12 – схема на свързване

### 3.1.1 Свързване на бредборд и захранващ модул

Захранването се поставя върху бредборда, като едната захранваща лента се настройва да е на  $+3.3V$  – за ESP-01 модула, а другата на  $+5V$  – за останалите компоненти. Един пиновете за земя на Ардуино платката се свързва с една от лентите за земя на бредборда, за да се постигне обща земя.

### 3.1.2 Свързване на DHT11

Сензорът за температура и влага DHT11 се свързва към бредборда и към Ардуино. VCC, Signal, GND се свързват съответно към +5V, +5V с  $10\text{K}\Omega$  издърпващ резистор към цифров пин на ардуиното и земя. Помощното захранване от +5V и резистор се използват за правилно отклоняване на входовете на цифровите логически устройства, за да не преминават от едно състояние в друг.

### 3.1.3 Свързване на реле

Релето се свързва директно към захранването (+5V) и земята съответно за VCC и GND крачетата. IN крачето приема цифров сигнал, чрез

който контролира външната схема, затова се свързва директно към цифров пин на Ардуино платката. Външната схема се свързва в режим на нормално отворена схема – NO и COM се използват.

### 3.1.4 Свързване на ESP-01

ESP8266-01 има 8 крачета, от които 5 са свързани към бредборда и Ардуино платката. VCC и GND са съответно свързани към +3.3V и земя на бредборда. CHPD крачeto, което се използва за стартирането на модула, се свързва към +3.3V с  $10\text{K}\Omega$  издърпващ резистор. Останалите две крачета се свързват към цифрови пинове на Ардуино платката, които се използват за серийна комуникация.

### 3.1.5 Описание на софтуерната част на проекта

## 3.2 Файлова структура на проекта

Проектът се състои от следните директории:

- *hardware* – файловете, които съдържат кода, който се изпълнява върху Ардуино платката
- *web-app* – всички файлове за уеб клиента, написан на *React.js*<sup>xiv</sup>
  - *src* – цялата функционалност и дизайн на уеб приложението
  - *public* – шаблонът, върху който се „рисува“ уеб приложението
- *server* – намира се целият код за *express.js* сървъра
- *doc* – намира се тази документация
- *assets*
  - *img* – всички снимки на проекта, както и използвани в *readme.md* файла в корена на тази файлова система
  - *schematics* – всички схеми на хардуерната част на проекта

## 3.3 Софтуерът за Ардуино платката

В началото на кода се намират библиотеките (фиг. 3.2а), използвани в този проект. Следвани са всички норми за писане на четлив код, като библиотеките са качени най-отгоре на *hardware.ino* файла. Библиотеките, които са използвани, са:

- *Ардуино JSON* – използвана за изпращане на данни като температура и влажност, като това улеснява процеса на клиентския софтуер, като пакетира данните в JSON обект.
- *DHT* и *DHT\_U* – библиотеките на DHT, които включват важните функции, с които се стартира работата на DHT11, както и да се чете информацията от него:
  - *DHT dht(DHTPIN, DHTTYPE);* и *dht.begin();* - функциите за инициализиране и стартиране на DHT11 или DHT22

- `dht.readHumidity();` и `dht.readTemperature();` - функциите за четене на влажност и температура от цифровите пинове
- Software Serial – е библиотека, която позволява серийна комуникация с цифров пин, различен от серийните пинове.

```
1 #include <ArduinoJson.h> // ArduinoJson 6+ only, btw -> works with the latest version of ArduinoJson
2 #include <DHT_U.h> // DHT library
3 #include <DHT.h> // DHT library
4 #include <SoftwareSerial.h> // SoftwareSerial library
```

*Фиг. 13 – добавяне на глобални библиотеки*

Собственоръчно направената локална библиотека *Ардуино\_secrets.h* (добавя се при *фиг. 12*) се използва за съхраняване на поверителна информация като Wi-Fi пароли, API ключове и други. В този проект се използва и за името на Wi-Fi мрежата и неговата парола.

```
1 #include "arduino_secrets.h" // Arduino secrets - Used to store Wi-Fi credentials
2 // [IMPORTANT]
3 // When cloning this project CREATE an "arduino_secrets.h" file with your Wi-Fi credentials
4 // [TEMPLATE] arduino_secrets.h
5 // [BEGINNING OF FILE]
6 // #define SSID "this is where you type your SSID"
7 // #define PASS "this is where you type your password"
8 // [END OF FILE]
```

*Фиг. 14 – добавяне на локалната библиотеката „arduino\_secrets.h“*

Отново спазвайки правилата за писане на чист и подреден код, всичките константи и/или глобални променливи отгоре на файла, но под библиотеките. Като тези глобални константни променливи са пиновете на релето и DHT11 както и типът на DHT сензорът – DHT11. Другите глобални променливи са Wi-Fi името и паролата, които се взимат от *Ардуино\_secrets.h*, както и променливите, които са полезни по време на отстраняване на грешки.

След тях се инициализира глобално комуникацията с ESP-01 (ESP8266), DHT11 сензорът и се създава JSON буфера, в който се ще се съхраняват данните за изпращане преди самото изпращане, като поддържа до 100 DHT сензора.

```

1 // DHT Config Variables
2 #define DHTPIN 7 // DHT digital pin number
3 #define DHTTYPE DHT11 // Type of the DHT sensor (in this case - DHT11) - values could be DHT11 or DHT22
4 // Relay Config
5 #define RELAY_PIN 12 // DHT relay digital pin number
6
7 // Wifi Credentials
8 String wifi_ssid = SSID;
9 String wifi_password = PASS;
10
11 // Debugging (Fail / Succ) with ESP8266
12 int countTrueCommand;
13 int countTimeCommand;
14 boolean found = false;
15
16
17 // Setting up global sensors x ESP8266
18 SoftwareSerial esp(3, 2);
19 DHT dht(DHTPIN, DHTTYPE);
20
21 // Buffer to store JSON object - ArduinoJSON 6+
22 StaticJsonDocument<200> root;

```

Фиг. 15 – глобални променливи и константи

Тук накратко може да се видят декларациите на функциите (извън *setup* и *loop*), които са дефинирани след тях, поради по-лесна четимост на кода.

```

1 // Function Declarations
2
3 // Func to set the esp up
4 void esp_setup();
5
6 // Function to determine success / failure of serial commands send to/from ESP8266
7 void sendCommandToesp(String command, int maxTime, char readReplay[]);
8
9 // Send HTTP Response from the Arduino and the ESP8266 to the client (aka the React / Flutter app)
10 // Func could also be used to SEND GET / POST requests
11 void sendData(String data);

```

Фиг. 16 – дефинициите на функции

В *setup()* (фиг. 15) се стартира всичко нужно за проекта – този код се изпълнява само веднъж при стартиране на Ардуино платката или при качването на нов код. Първоначално се инициализира пина на релето, като и се задава на стойността по подразбиране – *HIGH* (висока), тъй като релето се използва в режим на *NO* (нормално отворена връзка), защото компютрите се включват и изключват при натискане на бутон, което се зачита, като затворена връзка, за около 1 секунда. След това се инициализира DHT сензорът, а след него започва серийната комуникация на Ардуино платката и на ESP-01 на скорост на предаване (baud rate) от 115 200 – скоростта по

подразбиране на ESP модула. След това се извиква функцията *esp\_setup()* (фиг. 22), чрез която се стартира и настройва ESP-01 модула.

```
1 void setup() {  
2     // Setting the relay to its default state (just in case) - relay connected with NO (Normally Open)  
3     pinMode(RELAY_PIN, OUTPUT);  
4     digitalWrite(RELAY_PIN, HIGH);  
5  
6     // Setting up the DHT  
7     dht.begin();  
8  
9     // Serial Config (Arduino -> esp) - Baudrate @ 115 200  
10    Serial.begin(115200);  
11    esp.begin(115200);  
12  
13    // ESP Config  
14    esp_setup();  
15 }
```

Фиг. 17 – функцията *setup()*

В *loop()* функцията (фиг. 18) кодът, поставен в нея, се изпълнява многократно, докато Ардуино платката е включена..

Първоначално започва с проверка дали ESP-01 е налично и работещо (дали работи правилно се проверява в по-надолу в кода). След това се задава отговор по подразбиране при заявка на ардуиното от клиента (фиг. 18).

```
1 void loop() {  
2     if(esp.available()) {  
3         // Default response  
4         String def_res = "" + String("HTTP/1.1 200 OK\r\n") +  
5                         "Content-Type: none\r\n" +  
6                         "Content-Length: 0\r\n" +  
7                         "Access-Control-Allow-Origin: *\r\n" +  
8                         "Connection: close\r\n\r\n";  
9     ...
```

Фиг. 18 – началото на функцията *loop()*

Чрез следващите проверки (фиг. 19) „*if(esp.find("+IPD,") )*“ се проверява дали ESP-01 получава заявка от клиент. Ако е получило взема

номера на връзката и след това проверява каква команда му е дадена, като тя може да бъде една от следните 3:

```
1 ...
2     // Check if the ESP8266 is getting data
3     if(esp.find("+IPD,")) {
4         Serial.println("-----RECEIVED-----"); // [DEBUG]
5         delay(1000);
6         int connectionId = esp.read()-48; // Get Connection ID & convert it from ASCII to int
7
8         // Look for the command in the query string
9         if(esp.find("command=")){
10             int command = (esp.read()); // Get Command
11             delay(500);
12             Serial.println("COMMAND: " + command); // [DEBUG]
13             Serial.println("CONNID: " + connectionId); // [DEBUG]
14
15             // [LIST] Command List
16             // 1 -> Turn On / Off the computer
17             // 2 -> Send sensor data (temp & humidity) to the client
18             // 3 -> Force reset the computer (usually you have to short the power pins around 7 seconds to reset a computer)
19 ...
```

Фиг. 19 – вземане на параметри от url

- 1 – включване / изключване на компютъра – релето шунтира пиновете на дънната платка за 1.5 секунди и след това се изпраща *OK* на клиента (фиг. 20)

```
1 ...
2
3     if(command == '1'){
4         // Short the relay for 1.5 seconds
5         digitalWrite(RELAY_PIN, LOW);
6         delay(1500);
7         digitalWrite(RELAY_PIN, HIGH);
8
9         // Send HTTP Response to the client – to signal that the command was successfully executed
10        String cipSend = "AT+CIPSEND=" + String(connectionId) + "," + String(def_res.length());
11        sendCommandToesp(cipSend, 4, ">");
12        sendData(def_res);
13        String closeCommand = "AT+CIPCLOSE=";
14        closeCommand+=connectionId;
15        closeCommand+="\r\n";
16        sendCommandToesp(closeCommand, 5, "OK");
17
18 ...
```

Фиг. 20 – първа команда

- 2 – вземане на данни от сензора – температура и влажност – след това се изпращат данните към клиента (*фиг. 21*)

```

1 ...
2
3     } else if(command == '2') {
4         // Get data from the temp / humidity sensor via funcs form the DHT library
5         float h = dht.readHumidity();
6         float t = dht.readTemperature();
7
8         delay(500);
9
10        // Add the data to the JSON buffer
11        root["humidity"] = String(h);
12        root["temp"] = String(t);
13
14        // Stringify the JSON buffer
15        String data;
16        serializeJson(root, data);
17
18        // Send HTTP Response to the client - to send the data
19        String response = "" + String("HTTP/1.1 200 OK\r\n") +
20                    "Connection: close\r\n" +
21                    "Content-Length: " + data.length() + "\r\n" +
22                    "Content-Type: application/json\r\n" +
23                    "\r\n" + data;
24
25        String cipSend = "AT+CIPSEND=" + String(connectionId) + "," + String(response.length());
26        sendCommandToesp(cipSend, 4, ">");
27        sendData(response);
28        String closeCommand = "AT+CIPCLOSE=";
29        closeCommand+=connectionId;
30        closeCommand+="\r\n";
31        sendCommandToesp(closeCommand, 5, "OK");
32
33 ...

```

*Фиг. 21 – втора команда*

- 3 – принудително изключване – независимо от софтуерът на компютъра – релето шунтира пиновете на дънната платка за 7 секунди и след това се изпраща *OK* на клиента (*фиг. 22*)

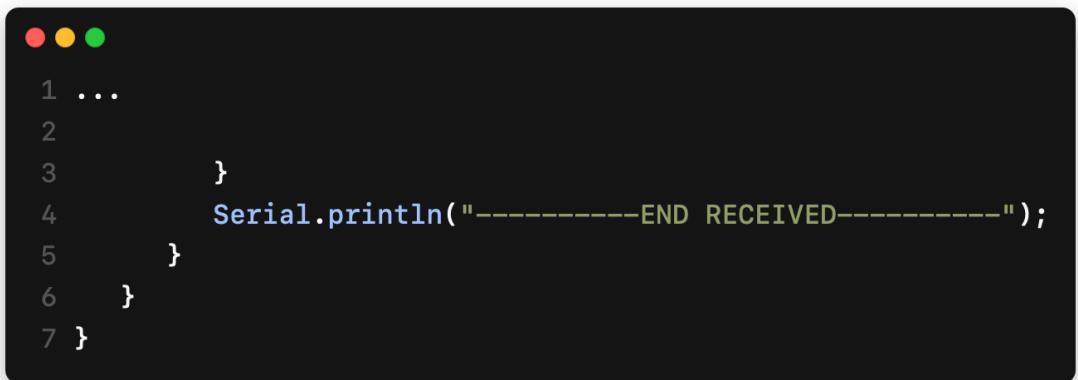
```

1 ...
2
3     } else if(command == '3'){
4         // Short the relay for 1.5 seconds
5         digitalWrite(RELAY_PIN, LOW);
6         delay(750);
7         digitalWrite(RELAY_PIN, HIGH);
8
9         // Send HTTP Response to the client - to signal that the command was successfully executed
10        String cipSend = "AT+CIPSEND=" + String(connectionId) + "," + String(def_res.length());
11        sendCommandToesp(cipSend, 4, ">");
12        sendData(def_res);
13        String closeCommand = "AT+CIPCLOSE=";
14        closeCommand+=connectionId;
15        closeCommand+="\r\n";
16        sendCommandToesp(closeCommand, 5, "OK");
17    }
18
19 ...

```

*Фиг. 22 – трета команда*

На фигура 23 приключва функцията *loop()*.



```
1 ...
2
3     }
4     Serial.println("-----END RECEIVED-----");
5 }
6 }
7 }
```

Фиг. 23 – край на *loop()*

В *esp\_setup()* (фиг. 24) функцията се нулира модулът, като след това се изчаква даден период от време. След този период започва работата на модула, поставя се в режим на работа като сървър и като клиент. Свързва се към Wi-Fi мрежа за достъп до интернет с вече създадените константи *wifi\_ssid* за име на мрежата и *wifi\_password* за парола на мрежата. Включва се в работа за множество връзки и се стартира сървър на порт 80 – по подразбиране за HTTP връзки.



```
1 // Function to set-up the ESP8266
2 void esp_setup(){
3     sendCommandToesp("AT+RST", 5, "OK"); // Reset the ESP8266 & wait for it to boot (set to 5 secs)
4     delay(1000); // Added delay for safety
5     sendCommandToesp("AT", 5, "OK"); // [TEST] Send AT command to ESP8266
6     sendCommandToesp("AT+CWMODE=3", 5, "OK"); // Set the ESP8266 to STA mode
7     sendCommandToesp("AT+CWJAP=\"" + wifi_ssid + "\",\"" + wifi_password + "\", 20, "OK"); // Connect to the WIFI network
8     sendCommandToesp("AT+CIPMUX=1\r\n", 5, "OK"); // Enable multiple connections
9     sendCommandToesp("AT+CIPSERVER=1,80\r\n", 5, "OK"); // Start the server on port 80
10 }
```

Фиг. 24 – функцията *esp\_setup()*

В *sendCommandToesp()* (фиг. 25) функцията се изпраща през сериината комуникация команда към ESP модула и се прави проверка дали тя е получена успешно.

```
1 // Function to send a command to the ESP8266
2 void sendCommandToesp(String command, int maxTime, char readReplay[]) {
3     Serial.print(countTrueCommand); // [DEBUG] Print the number of successful commands
4     Serial.print(". at command => "); // [DEBUG] text
5     Serial.print(command); // [DEBUG] Print the command
6     Serial.print(" "); // [DEBUG] text
7
8     while (countTimeCommand < (maxTime * 1)) {
9         esp.println(command); // Send command to the ESP8266
10        if (esp.find(readReplay)) { // Find the reply from the ESP8266
11            found = true;
12            break;
13        }
14
15        countTimeCommand++; // Timer for timeout - increment by one
16    }
17
18    // Here we signal if the command was received successfully or not
19    if (found == true) {
20        Serial.println("Success"); // [DEBUG] Print success message and up the counter of successful commands
21        countTrueCommand++;
22        countTimeCommand = 0;
23    } else if (found == false) {
24        Serial.println("Fail"); // [DEBUG] Print error message and reset the counter
25        countTrueCommand = 0;
26        countTimeCommand = 0;
27    }
28
29    found = false;
30 }
```

Фиг. 25 – функцията *sendCommandToesp()*

Чрез функцията *sendData()* (фиг. 26) се изпраща през серийната комуникация данни към ESP модула.

```
1 // Function to send data to the client (aka the React / Flutter app)
2 void sendData(String data) {
3     Serial.println(data); // [DEBUG] Print the data to the serial monitor
4     esp.println(data); // Send the data to the client
5     delay(1500);
6     countTrueCommand++;
7 }
```

Фиг. 26 – функцията *sendData()*

## 3.4 Софтуерът за сървъра

### 3.4.1 Уеб клиент

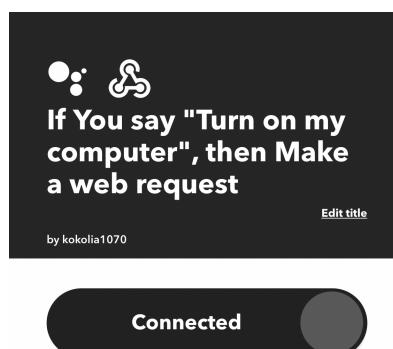
Сървърът за уеб клиентът е написан на популярния фреймуърк на node.js<sup>xv</sup> express.js<sup>xvi</sup>, поради неговата улесненост на писане и документация. Този приложно-програмен интерфейс<sup>xvii</sup> има за цел да

получи данните от ESP-01 сървъра и да ги предаде безопасно (с CORS<sup>xviii</sup>) на уеб клиента.

### 3.4.2 Гласов клиент

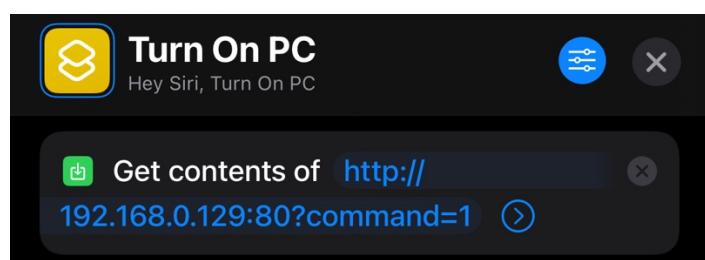
За гласовият клиент са използвани приложно-програмните интерфейси на IFTTT (фиг. 27) и Shortcuts (фиг. 28-30) на Apple, поради техните екосистеми. IFTTT се поддържа от 2 от най-големите гласови асистенти – Google Assistant и Alexa. Докато Shortcuts се използва при гласовия асистент на Apple – Siri.

За да работи IFTTT интерфейса му се задават входна и изходна команда (фиг. 27). Като входа идва от потребителя на гласовия асистент – с ключов израз „Turn on my computer”, който може също да се интерпретира и по различни начини като „Turn on my PC”, „Power on my computer“ и т.н. А за изходна – уеб заявка към ESP-01 сървъра – като се подадат IP адрес и параметри. За IP адрес се използва публичния адрес – към рутера, който е свързан ESP-01 модула и отворения му порт.

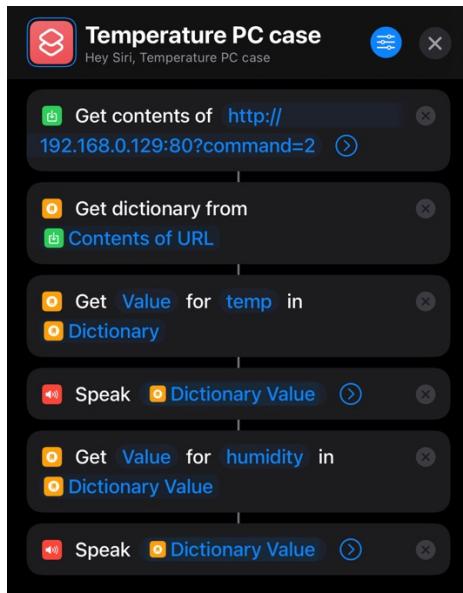


Фиг. 27 - IFTTT

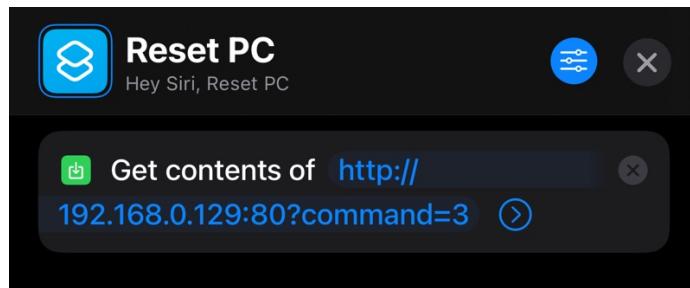
Интерфейсът Shortcuts работи на подобен принцип само че на локално ниво – изпълнява се на устройството (фиг. 28 – 30)



Фиг. 28 – Shortcuts – начин на изпълнение на команда 1



Фиг. 29 – Shortcuts – начин на изпълнение на команда 2

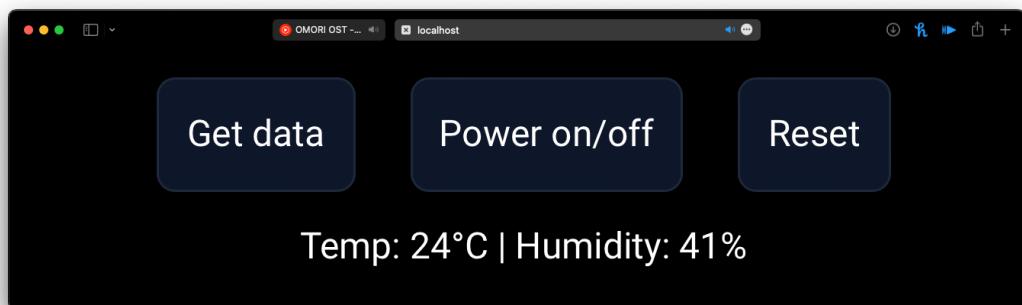


Фиг. 30 – Shortcuts – начин на изпълнение на команда 3

### 3.5 Уеб клиента и гласовите клиенти

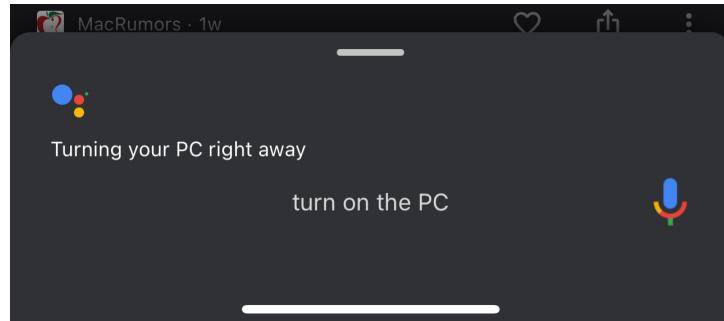
Уеб клиента – приложението (фиг. 31) е с лек и прост дизайн, направен да работи на всякакво устройство при каквато и да е интернет връзка. На потребителят са показани само три бутона – Get data, Power on/off и Reset.

Натискайки бутона за взимане на данни, потребителят получава текущите температура и влага на дадена машина под бутоните. А при останалите два бутона – се изпълняват кореспондирящите команди – за включване / изключване или принудително изключване.

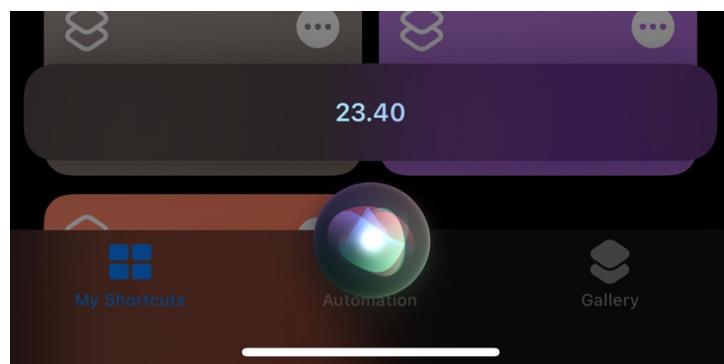


Фиг. 31 – уеб клиент

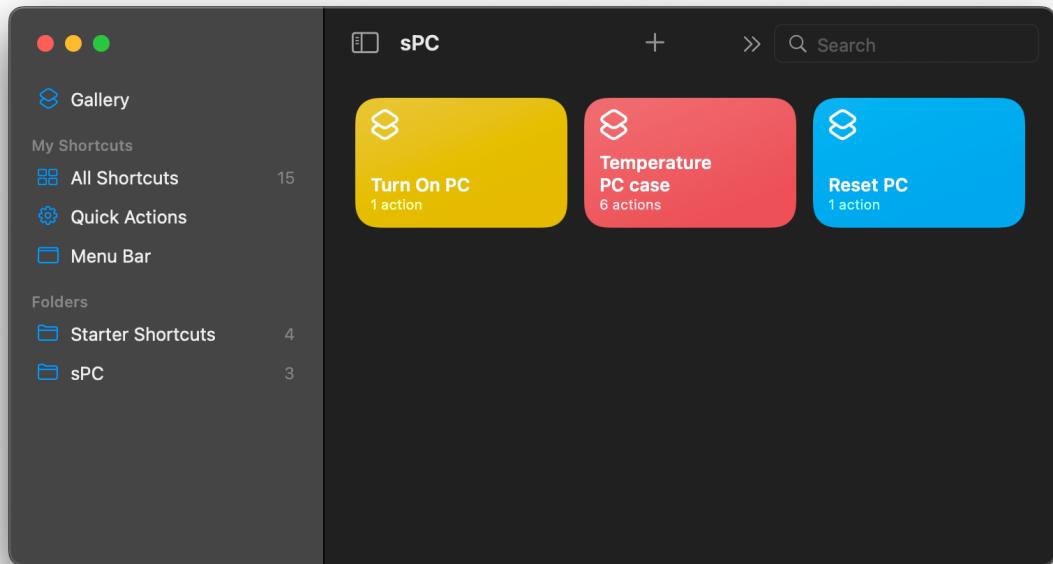
Потребителят може и да не използва уеб приложението, а вместо това – гласов асистент по негово предпочтение – Google Assistant, Alexa или Siri. Като това са най-използваните гласови асистенти на пазара за Android (фиг. 32), iOS (фиг. 33) и macOS (фиг. 34).



Фиг. 32 – Google Assistant при командата „Turn on the PC“



Фиг. 33 – Siri при командата „What's the temperature in computer's case?“



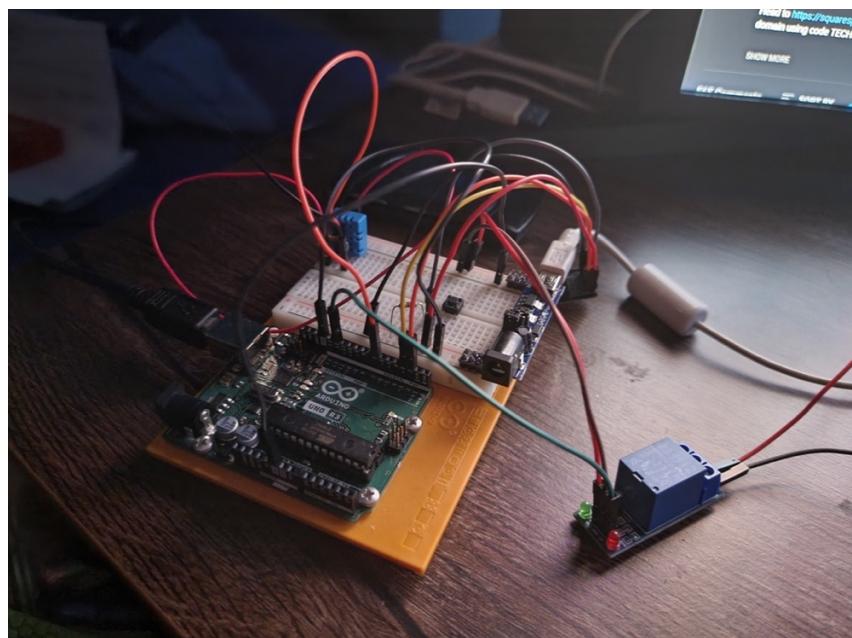
Фиг. 34 – Shortcuts приложението на macOS, показвайки трите команди за проекта

## ЧЕТВЪРТА ГЛАВА

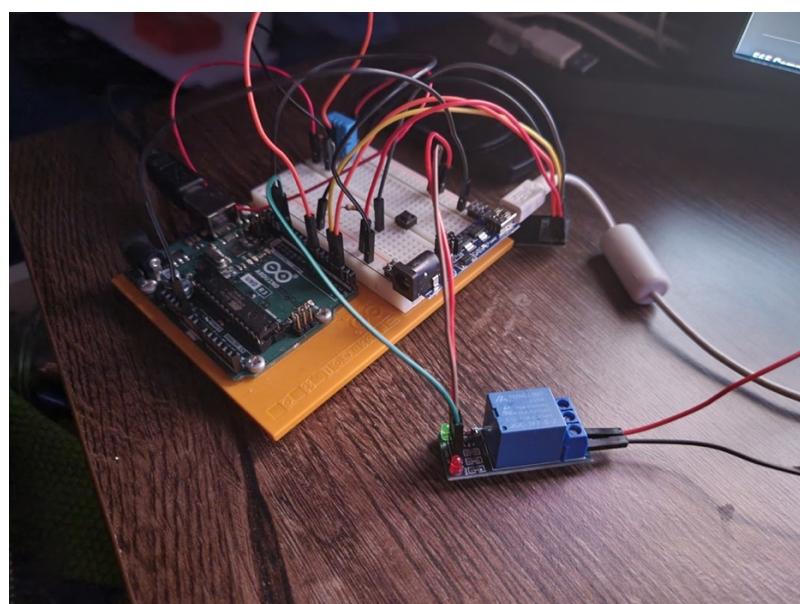
# СЪЗДАВАНЕ НА РАБОТОСПОСОБЕН МОДЕЛ НА ПРОЕКТА

### 1. Създаване на работоспособен модел на проекта

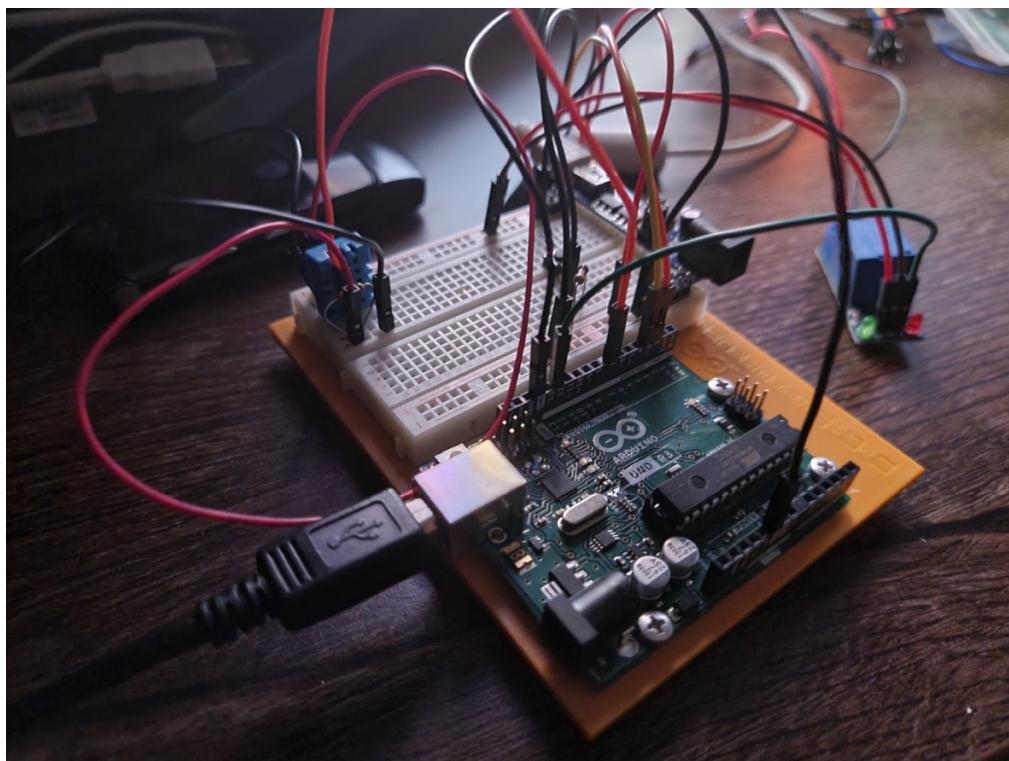
Снимки на хардуерната част от проекта може да се видят на *фигури от 31 до 34.*



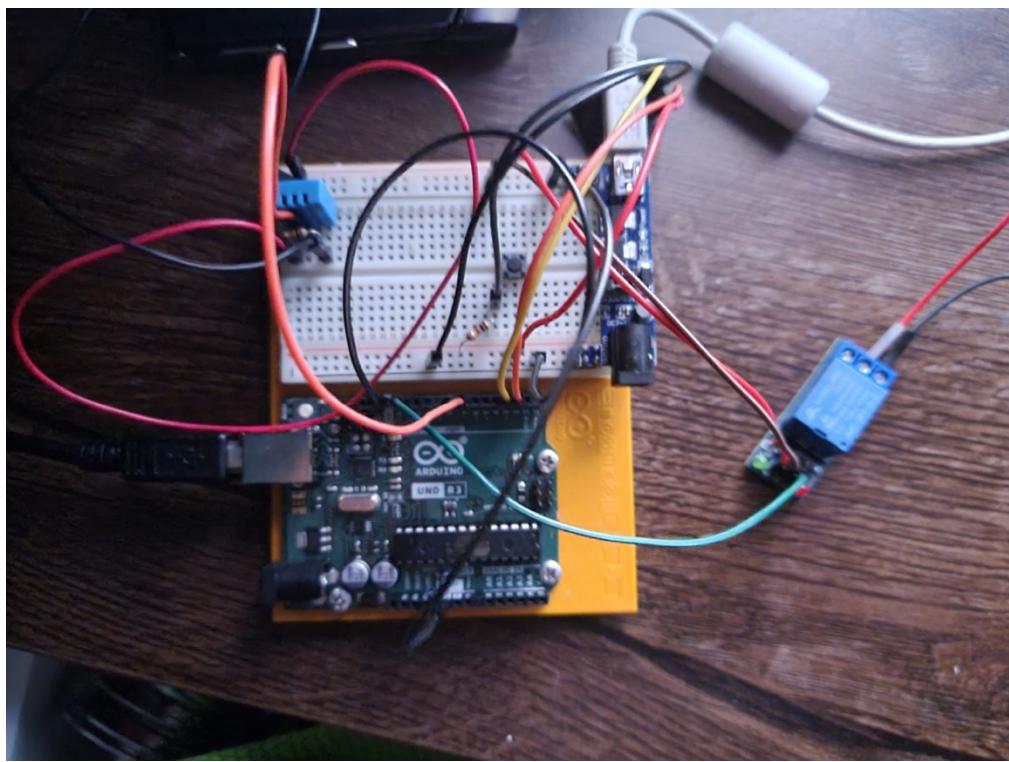
Фиг. 31



Фиг. 32



Фиг. 33



Фиг. 34

## **ЗАКЛЮЧЕНИЕ**

В описаната курсова работа са изпълнени всички заложени изисквания, включително функционални изисквания към хардуерна реализация, софтуерна реализация и разработено устройство. Създадена е работоспособна система. Бъдещото развитие на проекта се състои от създаването на документация и начин за разработване на английски език на Ардуино.cc и добавянето на звук за сензор, чрез който ще може да се наблюдава колко звук издават вентилаторите на даден компютър или сървър.

## ИЗПОЛЗВАНА ЛИТЕРАТУРА

---

- <sup>i</sup> DIY – Do It Yourself – собственоръчно направени
- <sup>ii</sup> Кратка информация за продуктите - <https://elimex.bg>
- <sup>iii</sup> Arduino Uno - <https://docs.arduino.cc/hardware/uno-rev3>
- <sup>iv</sup> Arduino Nano - <https://docs.arduino.cc/hardware/nano>
- <sup>v</sup> DHT11 и DHT22 – <https://www.adafruit.com/product/386>,  
<https://randomnerdtutorials.com/complete-guide-for-dht11dht22-humidity-and-temperature-sensor-with-arduino/>
- <sup>vi</sup> ESP - <https://www.esp8266.com>
- <sup>vii</sup> SoC – System on a Chip - Едночипова система
- <sup>viii</sup> Кит K2227 захранващ модул за бредборд -  
<https://elimex.bg/product/85003-kit-k2227-zahranvasht-modyl-za-bredbord>
- <sup>ix</sup> API на английски -  
<https://support.paysera.com/index.php?/payserabgr/Knowledgebase/ArticleView/1953/228/1903-kakvo-e-api>
- <sup>x</sup> IFTTT - <https://ifttt.com>
- <sup>xi</sup> Google Assistant - <https://assistant.google.com>
- <sup>xii</sup> Shortcuts - <https://support.apple.com/guide/shortcuts/welcome/ios>
- <sup>xiii</sup> Siri - <https://www.apple.com/siri/>
- <sup>xiv</sup> React.js – библиотека на JS - <https://reactjs.org>
- <sup>xv</sup> Node.js - <https://nodejs.org/en/>
- <sup>xvi</sup> express.js - <https://expressjs.com>
- <sup>xvii</sup> API на английски
- <sup>xviii</sup> CORS – <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>