**I. Съдържание**

I. Съдържание  
II. Увод  
III. Преглед и подбор на елементната база  
 1. Стара кафемашина

2.Arduino Mega

3. Solid state реле SSR-40 DA

4.DS3231 Real Time Clock

5.ТМ1637 display

6.ESP-01 WiFi Module

7.ESP-01 USB Programmer

8.Micro SD Card Module

9.2 броя бутони

10.ML144110VP 6 bit DAC

11.UA741 Op-Amp

12.Схема за засичане на нивото на водата  
IV. Реализация на устройството  
 1. Блок схеми на Hardware  
 2. Блок схеми на Software   
 3. Принципна електрическа схема

4. Софтуер - Код  
V. Същинска реализация  
 1. Снимки   
 2. Трудности   
 3. Успехи и неуспехи  
VI. Икономически аспект  
VII. Бъдещо развитие

**II. Увод**

В днешно време хората водят забързан начин на живот. Всяка секунда е ценна за тях. Като погледнем света, навсякъде хората се опитват да автоматизират всекидневните си дейности. Нашият проект прави точно това.

Всеки човек, в някой момент от живота си, иска да се наслади на чаша освежаващо кафе. Но не всеки има времето за това удоволствие. Затова ние разработихме кафемашина, която да ни спести това време. Нашата кафемашина се пуска автоматично, като ни известява със звуков сигнал, когато кафето ни е готово. Така тя ни спестява време и същевременно се събуждаме с приятният аромат на кафе.

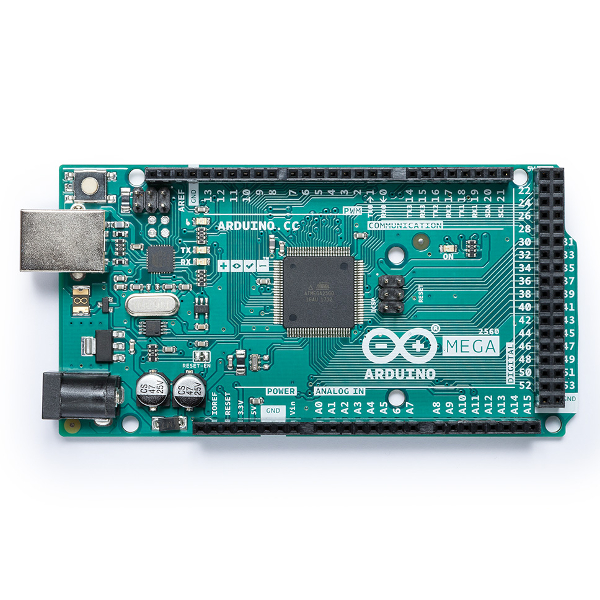
Освен това кафемашината ни може да бъде пусната дистанционно чрез мобилно устройство или компютър.

**III. Преглед и подбор на елементната база**

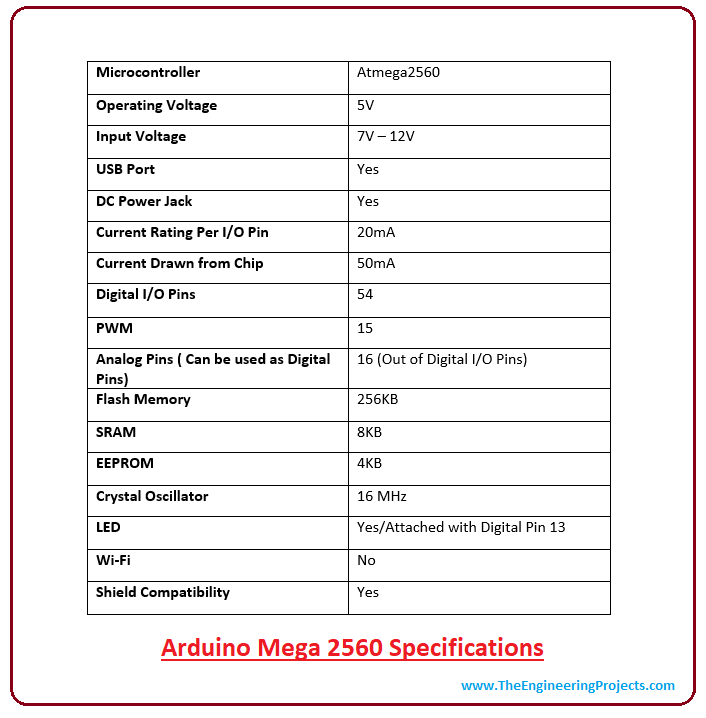
1. **Стара кафемашина .** Нашият проект използва стара кафемашина, която по този начин се рециклира. В случая използваме Кaфеварка Rowenta FG 262. Тя работи на променливо напрежение – 230V, честота 50Hz и мощност 800W.



**2.Arduino Mega -** микроконтролерна развойна платка изградена с AТmega2560 AVR MCU. Има 54 цифрови входно-изходни (I/O) порта, 16 аналогови входа, 4 UARTs порта (hardware serial ports), 16 MHz резонатор, четири светодиода (един потребителски, свързан на 13ти цифров I/O порт и три, които индикират работата на платката: ON, Tx и Rx), USB конектор, захранващ куплунг, бутон за рестартиране и два ICSP конектора (по един за ATmega2560 и ATmega16U2). 15 от цифровите I/O порта могат да се използват като PWM (ШИМ) изходи. Свързването с компютър се осъществява чрез USB кабел - USB A to USB B.



Характеристики:



1. **Solid state реле SSR-40 DA**

SSR полупроводниково реле с общо предназначение, еднофазно с максимално комутационно напрежение 380 VAC, максимален ток на товара 40 A и управляващо входно напрежение от 3 VDC до 32 VDC.



Характеристики:

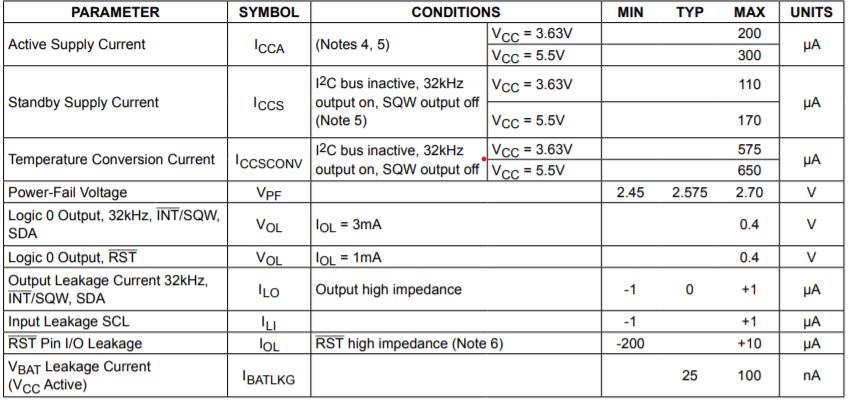
* Модел: SSR-40 DA;
* Серия на релето: SSR;
* Тип реле: Полупроводниково;
* Версия на релето: Монофазно;
* Управляващо напрежение: от 3 VDC до 32 VDC;
* Максимален ток на изхода (в зависимост от работната температура на корпуса на SSR за активен товар): 40 A;
* Номинално напрежение на комутация: 380 VAC;
* Бързодействие: 10 ms;
* Монтаж на свързващите проводници: Механично чрез винтове М4 mm за входа и М4 mm за изхода с кръстата отвертка;
* Монтаж на захващането: Два отвора Ø4.5 mm;
* Разстояние между монтажните отвори: 47.7 mm;
* Индикация на работното състояние: LED;
* Работна температура: -30 °C до +75 °C;
* Съпротивление на изолацията: 1000 Mohm / 500 VDC;
* Максимално пиково напрежение: 2500 VAC за 1 минута;
* Предпазен капак: Прозрачна пластмаса;

**4.DS3231 Real Time Clock**

DS3231 е прецизен RTC clock модул за Arduino. Има вградена батерия и запазва състоянието си дори ако има прекъсване на захранването.

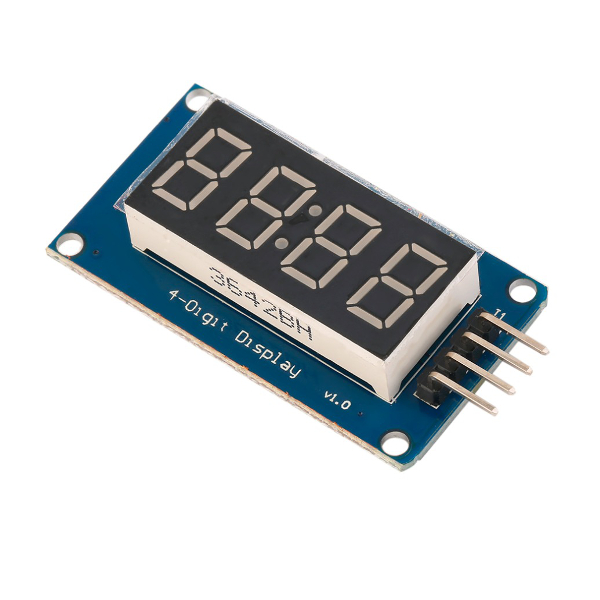


Характеристики:



**5.ТМ1637 display**

Модулът е изграден от четворен 7-сегментен светодиоден индикатор и LED драйвер TM1637. Има цифров интерфейс и се управлява само с два цифрови порта на микроконтролер. Захранва се и работи с напрежение 3V - 5,5V, и консумира приблизително 27mA ток.

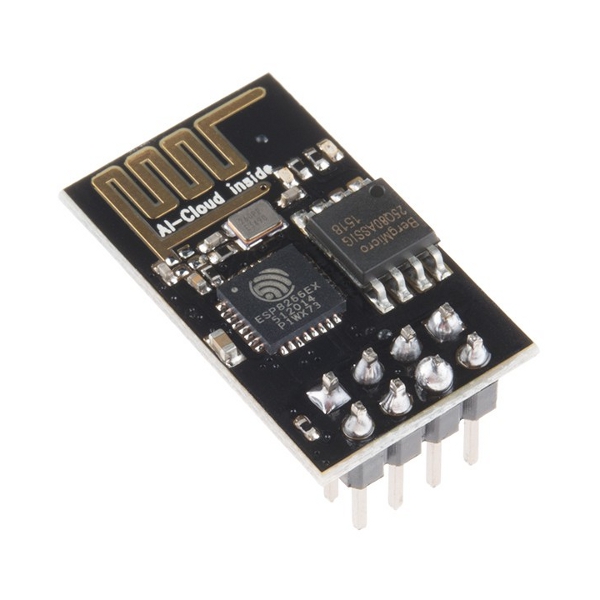


Характеристики:

* Размери: 42mm x 24mm x 14mm
* Тегло: 7 гр
* Захранване: 3V - 5,5V
* Консумация на ток: 27mA при 5V, максимално 80mA
* Работно напрежение: 3V - 5,5V
* Индикатор: светодиоден - червен, 7-сегментен, четворен, с 2 точки в средата - полезни при използване като часовник
* Драйвер: TM1637
* Интерфейс: цифров

**6.ESP-01 WiFi Module**

Модулът е изграден с WiFi 32-битов микрoконтролер ESP8266 . Има сериен интерфейс, два GPIO порта, допълнителна Flash памет и вградена антена. Платката е с размери 25mm x 15mm, с монтиран от долната ѝ страна интерфейсен конектор, на който са изведени портовете за захранване и комуникация.



Характеристики:

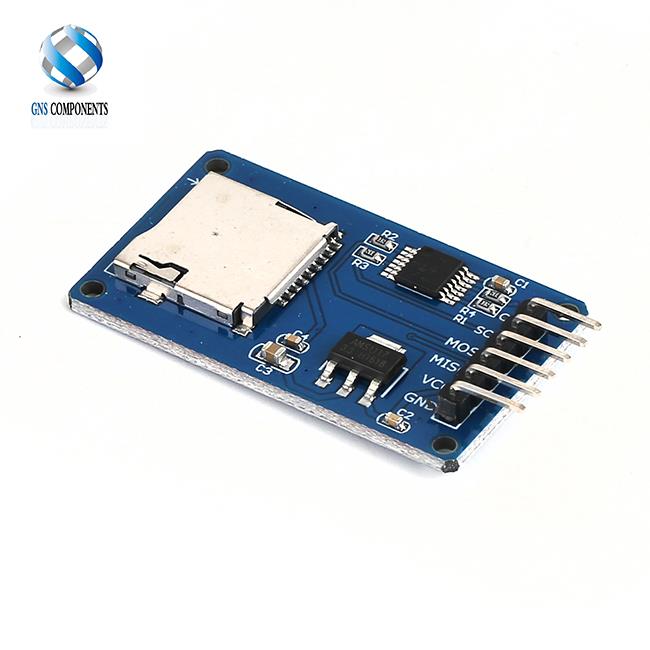
* + Микроконтролер: ESP8266, 32-bit, 802.11 b/g/n, Wi-Fi Direct (P2P), soft-AP
  + Flash: 1MB
  + Размери: 25mm x 15mm
  + Тегло: 1,4 г
  + Захранване: 3,3V
  + Работно напрежение: 3,3V, не толерира 5V
  + Консумация на ток: стандартно 60mA, максимално 220mA
  + Консумация на ток - Standby: 0,9uA
  + Вградена антена
  + Интерфейс: сериен UART, baud 115200
  + Зареден с фърмуер за управление с AT команди
  + Може да се програмира с Arduino IDE
  + Може да се зареди и работи с Lua фърмуер, MicroPython и др.
* **7.ESP-01 USB Programmer**

USB Модулът е предназначен за употреба с WiFi ESP-01 (ESP8266-01) модул, като за целта на платката е поставена двойна рейка (8 pins - 2x4P). WiFi модула се поставя върху USB конвертора (програматора) спазвайки ориентацията и заедно се включват в USP порт на компютъра. Така устройството е готово за работа използвайки AT команди. За програмирането на ESP-01 WiFi модула (използвайки Arduino IDE) първо трябва да свържете GPIO0 със GND, и тогава да поставите в USB порта. Модула се захранва от USB порта на компютъра и преобразува напрежението на 3,3 волта за WiFi модула.



**8.Micro SD Card Module**

Модулът (MicroSD Card Adapter) е модул за четене и запис на Micro SD карта. Чрез файловата система и драйвер за интерфейс SPI системата MCU може да завърши четенето и писането на файлове в MicroSD картата. Потребителите на Arduino могат директно да използват библиотеката на SD карти, която се предлага с Arduino IDE, за да инициализират и прочетат картата.



Характеристиките на модула са както следва

1. Поддръжка на Micro SD карта, Micro SDHC карта (карта с висока скорост)

2. Вградена верига за изместване на ниво на борда, т.е. нивото на интерфейса може да бъде 5V или 3.3V

3. Захранването е 4.5V ~ 5.5V, вградена 3.3V регулаторна верига

4. Комуникационният интерфейс е стандартен SPI интерфейс

5. 4 отвора за позициониране на винт M2 за лесна инсталация

Интерфейсни параметри

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| проект | Минимална стойност | Типична стойност | Максимална стойност | Мерна единица |
| Захранващо напрежение VCC | 4.5 | 5 | 5.5 | V |
| Текущ | 0.2 | 80 | 200 | mA |
| Интерфейсно ниво | 3.3 или 5 | | | V |
| Тип карта за поддръжка | Micro SD карта (<= 2g),="" mirco="" sdhc="" карта=""><=> | | | - |
| размер | 42X24X12 | | | mm |
| тегло | 5 | | | г |

**9. 2 броя бутони**

Бутоните са предназначени за монтаж на печатна платка, но могат да се изполават и на прототипна платка breadboard.



Спецификация:

Максимални стойности: DC 12 V , 50 mA , 100000 натискания

**10.ML144110VP 6 bit DAC**

Поке добави тук

**11.UA741 Op-Amp**

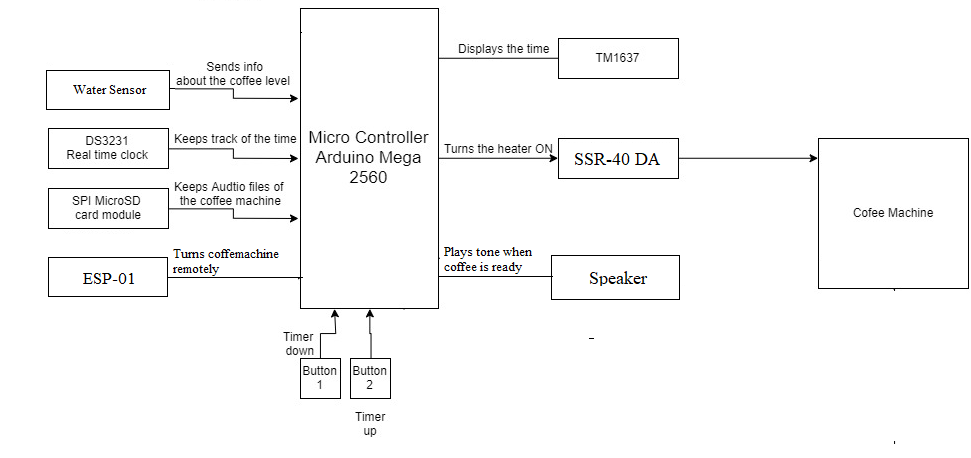
Поке добави тук

**12.Схема за засичане на нивото на водата**

Поке добави тук

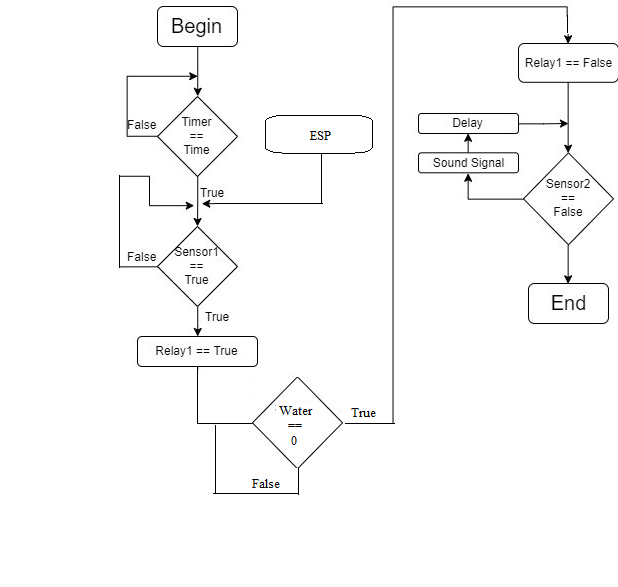
**IV. Реализация на устройството**

1. **Блок схема на Хардуера**



Часът се задава с бутони 1 и 2 и се изобразява на TM11637. DS3231 следи часа. Сензорът за вода следи нивото на водата, за да може да се спре кафемашината. Файловете се четат от SD картата и се възпроизвеждат на колонката. През ESP модула се пуска кафемашината отдалечено.

1. **Блок схема на Software**



1. **Принципна електрическа схема**

Поке добави тук

1. **Софтуер - Код  
   Ардуино:**

|  |
| --- |
| #include <DS3231.h> |
|  | #include <TM1637Display.h> |
|  | // Display pins |
|  | #define CLK 9 |
|  | #define DIO 28 |
|  | // Create display object of type TM1637Display: |
|  | TM1637Display display = TM1637Display(CLK, DIO); |
|  |  |
|  | // Init the DS3231 using the hardware interface |
|  | DS3231 rtc(SDA, SCL); |
|  |  |
|  | //water sensor pin |
|  | int waterPin = 33; |
|  | //esp pin |
|  | const int espPin = 32; |
|  | //relay pins |
|  | const int relayPin = 10; |
|  | //button pins |
|  | int but1Pin = 31; |
|  | int but2Pin = 30; |
|  | int CupButton = 45; |
|  |  |
|  | //water sensor variables |
|  | unsigned long duration = 10000; |
|  | //button variables |
|  | int buttonPushCounter1 = 0; // counter for the number of button presses |
|  | int buttonState1 = 0; // current state of the button |
|  | int lastButtonState1 = 0; // previous state of the button |
|  |  |
|  | int buttonPushCounter2 = 0; // counter for the number of button presses |
|  | int buttonState2 = 0; // current state of the button |
|  | int lastButtonState2 = 0; // previous state of the button |
|  |  |
|  | //clock time |
|  | int timerHours = 17; |
|  | int timerMinutes = 0; |
|  | void setup() |
|  | { |
|  | // Setup Serial connection |
|  | Serial.begin(115200); |
|  |  |
|  | // IStarts the clock module |
|  | rtc.begin(); |
|  |  |
|  | //sets relay pins |
|  | pinMode(relayPin, OUTPUT); |
|  | //sets waterSensorPins |
|  | pinMode(waterPin, INPUT); |
|  | //sets buttons pins |
|  | pinMode(but1Pin, INPUT); |
|  | pinMode(but2Pin, INPUT); |
|  | pinMode(CupButton, INPUT); |
|  | //sets espPin |
|  | pinMode(espPin, INPUT); |
|  | //sets default value to off to relays |
|  | digitalWrite(relayPin, LOW); |
|  | // Clear the display: |
|  | display.clear(); |
|  | // The following lines can be uncommented to set the date and time |
|  | rtc.setDOW(SUNDAY); // Set Day-of-Week to SUNDAY |
|  | rtc.setTime(17, 05, 00); // Set the time to 12:00:00 (24hr format) |
|  | rtc.setDate(29, 12, 2019); // Set the date to January 1st, 2014 |
|  | } |
|  |  |
|  | void loop() |
|  | { |
|  | // Set the brightness: |
|  | display.setBrightness(7); |
|  |  |
|  |  |
|  |  |
|  | //divide by ":" to get the time in int |
|  | char \*buf = rtc.getTimeStr(); |
|  | int i = 0; |
|  | char \*p = strtok (buf, ":"); |
|  | char \*arr[3]; |
|  | while (p != NULL) |
|  | { |
|  | arr[i++] = p; |
|  | p = strtok (NULL, ":"); |
|  | } |
|  | //set hours and minutes |
|  | int hours = atoi(arr[0]); |
|  | int minutes = atoi(arr[1]); |
|  |  |
|  |  |
|  | // Print time on serial monitor just in case |
|  | Serial.println(rtc.getTimeStr()); |
|  | //hours = hours\*100 + minutes; |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | /\* |
|  | \* TEST 1 |
|  | // Print 1234 with the center colon: |
|  | display.showNumberDecEx(hours, 0b11100000, false, 4, 0); |
|  |  |
|  | if(distance <= 5 && hours >= 1505){ |
|  | digitalWrite(relayPin, LOW); |
|  | }else if(distance > 5 && hours >= 1505){ |
|  | digitalWrite(relayPin, HIGH); |
|  | } |
|  | \*/ |
|  |  |
|  | //Button1 timer++ |
|  | buttonState1 = digitalRead(but1Pin); |
|  | if (buttonState1 != lastButtonState1) { |
|  | if (buttonState1 == HIGH) { |
|  | timerMinutes++; |
|  | if(timerMinutes >= 60){ |
|  | timerMinutes = 0; |
|  | timerHours++; |
|  | if(timerHours >= 24){ |
|  | timerHours = 0; |
|  | } |
|  | } |
|  | } |
|  | delay(50); |
|  | } |
|  | lastButtonState1 = buttonState1; |
|  |  |
|  | //Button2 timer-- |
|  | buttonState2 = digitalRead(but2Pin); |
|  | if (buttonState2 != lastButtonState2) { |
|  | if (buttonState2 == HIGH) { |
|  | timerMinutes--; |
|  | if(timerMinutes < 0){ |
|  | timerMinutes = 59; |
|  | timerHours--; |
|  | if(timerHours < 0){ |
|  | timerHours = 23; |
|  | } |
|  | } |
|  | } |
|  | delay(50); |
|  | } |
|  | lastButtonState2 = buttonState2; |
|  |  |
|  | int timerOnDisplay = timerHours\*100 + timerMinutes; |
|  | display.showNumberDecEx(timerOnDisplay, 0b11100000, false, 4, 0); |
|  |  |
|  | //logic |
|  | if((hours == timerHours && minutes == timerMinutes) || (digitalRead(espPin) == HIGH)){ |
|  | if(digitalRead(CupButton) == HIGH){ |
|  | delay(2000); |
|  | if(digitalRead(CupButton) == HIGH){ |
|  | digitalWrite(relayPin, HIGH); |
|  | duration = pulseIn(waterPin, HIGH); |
|  | Serial.println("I am making cofee!!!"); |
|  | while(duration <= 900){ |
|  | duration = pulseIn(waterPin, HIGH); |
|  | Serial.println(duration); |
|  | } |
|  | digitalWrite(relayPin, LOW); |
|  | int seconds\_counter = 0; |
|  | while(digitalRead(CupButton) == HIGH){ |
|  | delay(1000); |
|  | seconds\_counter++; |
|  | if(seconds\_counter == 10){ |
|  | Serial.println("Your coffee will get cold!"); |
|  | } |
|  | if(seconds\_counter == 20){ |
|  | Serial.println("Why did I make this coffee!"); |
|  | } |
|  | if(seconds\_counter >= 30){ |
|  | Serial.println("Take your coffee!"); |
|  | delay(1000); |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  | } |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  | // Wait one second before repeating :) |
|  | //delay (1000); |
|  | } |

**ESP:**

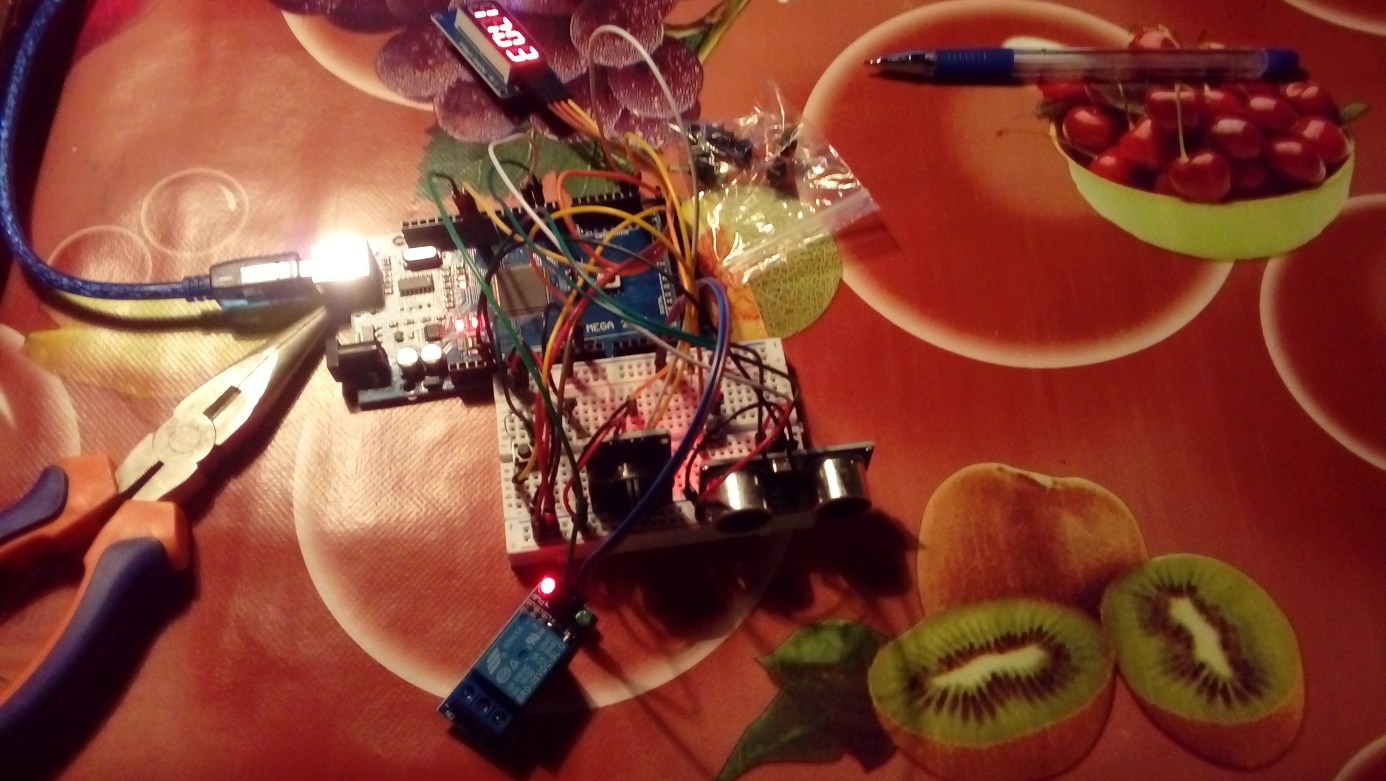
|  |
| --- |
| #include <ESP8266WiFi.h> |
|  |  |
|  |  |
|  | const char\* ssid = "Coffee-Machine"; |
|  | const char\* password = "parolata"; |
|  |  |
|  | const int pin = 2; |
|  |  |
|  | WiFiServer server(80); |
|  |  |
|  | void setup() |
|  | { |
|  |  |
|  | Serial.begin(115200); |
|  | Serial.print("Setting soft-AP ... "); |
|  | boolean result = WiFi.softAP(ssid, password); |
|  | if(result == true) |
|  | { |
|  | Serial.println("Ready"); |
|  |  |
|  |  |
|  | server.begin(); |
|  | Serial.printf("Web server started, open %s in a web browser\n", WiFi.localIP().toString().c\_str()); |
|  | } |
|  | else |
|  | { |
|  | Serial.println("Failed!"); |
|  | } |
|  |  |
|  | pinMode(pin,OUTPUT); |
|  | } |
|  |  |
|  | // prepare a web page to be send to a client (web browser) |
|  | String prepareHtmlPage() |
|  | { |
|  | String htmlPage = |
|  | String("HTTP/1.1 200 OK\r\n") + |
|  | "Content-Type: text/html\r\n" + |
|  | "Connection: close\r\n" + // the connection will be closed after completion of the response |
|  | "\r\n" + |
|  | "<!DOCTYPE HTML>" + |
|  | "<html>" + |
|  | "<body> <h1>Coffee Machine Home Page</h1>" + |
|  | "<form action=\"/gpio2/1\" method=\"GET\"><input type=\"submit\" value=\"ON\"></form><form action=\"/gpio2/0\" method=\"GET\"><input type=\"submit\"value=\"OFF\"></form>"+ |
|  | " </body></html>" + |
|  | "\r\n"; |
|  | return htmlPage; |
|  | } |
|  |  |
|  |  |
|  | void loop() |
|  | { |
|  | WiFiClient client = server.available(); |
|  | // wait for a client (web browser) to connect |
|  | if (client) |
|  | { |
|  | Serial.println("\n[Client connected]"); |
|  | while (client.connected()) |
|  | { |
|  | // read line by line what the client (web browser) is requesting |
|  | if (client.available()) |
|  | { |
|  | String line = client.readStringUntil('\r'); |
|  | Serial.print(line); |
|  | // wait for end of client's request, that is marked with an empty line |
|  | if (line.length() == 1 && line[0] == '\n') |
|  | { |
|  | if (line.indexOf("/gpio2/1") != -1){ |
|  | digitalWrite(pin,1); |
|  | }else if (line.indexOf("/gpio2/0") != -1){ |
|  | digitalWrite(pin,0); |
|  | } |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | delay(1); // give the web browser time to receive the data |
|  |  |
|  | // close the connection: |
|  | client.stop(); |
|  | Serial.println("[Client disonnected]"); |
|  | } |
|  | } |

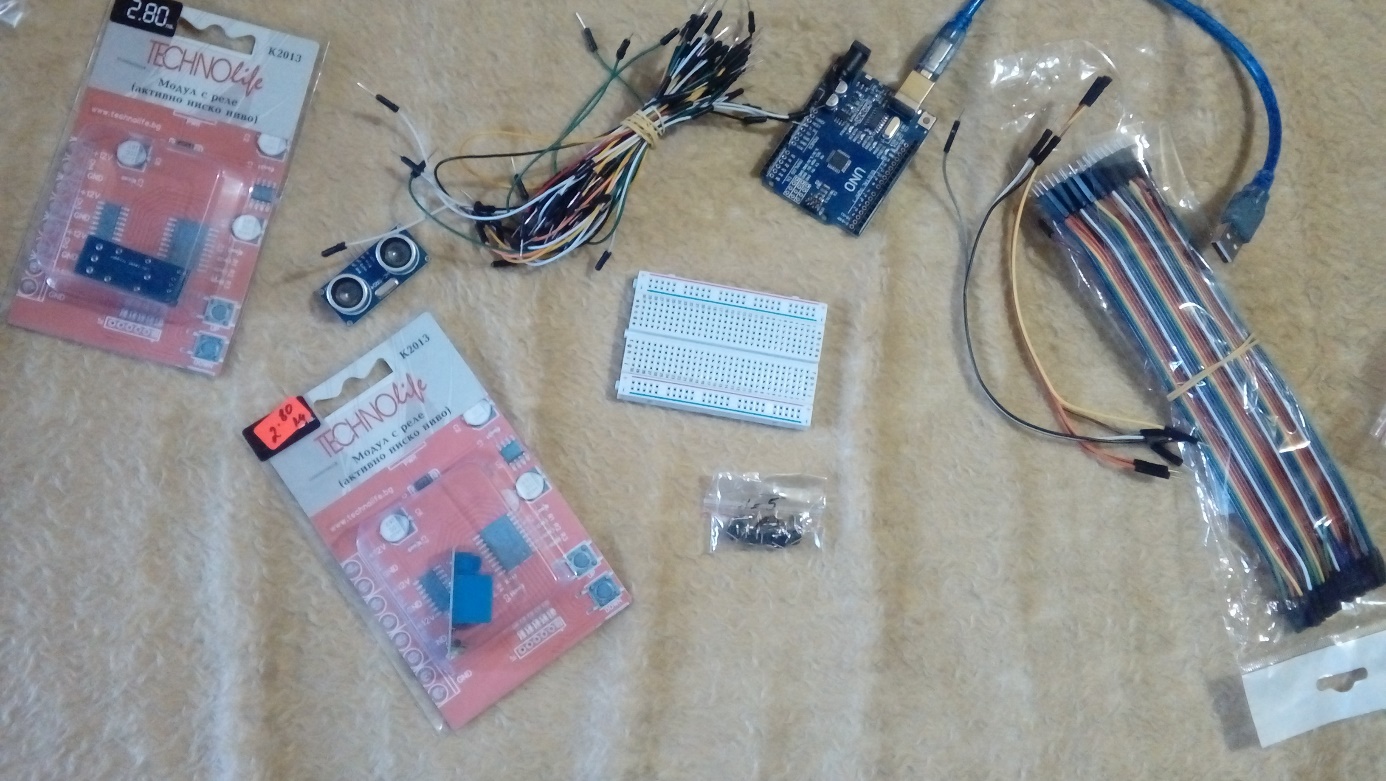
**V. Същинска реализация**

**1. Снимки**









1. **Трудности при изработката на проекта**

По време на разработката на проекта се срещнахме с редица трудности:

* + Първоначалната кафемашина имаше прекалено сложна схема и трябваше да я заменим с друга
  + Релетата, които бяхме взели бяха прекалено слаби и трябваше да купим по мощни
  + Ултразвуковият сензор, с който смятахме да определяме нивото отчиташе грешни стойности, заради влагата.
  + Програмирането на ESP-то
  + Комуникацията и размяната на части бяха почти невъзможни по време на карантината
  + Заснемането на клипа беше трудно, заради ограниченото време
  + Нагревателят е прекалено силен и трябваше да извадим всички елементи извън корпуса на кафемашината, защото щяха да прегреят

1. **Успехи и неуспехи**

Успешно са реализирани:

* + Подбор на елементната база
  + Настройване на времето на пускане
  + Пускане на машината в определен час
  + Пускане на машината отдалечено
  + Засичане на нивото на водата
  + Свързаност на релето с машината
  + Засичане на наличието на кана
  + Побиране на електроника и правилно окабеляване + cable management
  + Възпроизвеждане на единичен тон

Неуспехи:

* + Реализация на възпроизвеждане на цели аудио файлове
  + Съединяване на двете части от проекта

**VI. Икономически Аспект**

1. Стара кафемашина – 20лв

2.Arduino Mega – 20 лв

3. Solid state реле SSR-40 DA – 7.24лв

4.DS3231 Real Time Clock – 8лв

5.ТМ1637 display – 4.30лв

6.ESP-01 WiFi Module – 6.60лв

7.ESP-01 USB Programmer – 4.80лв

8.Micro SD Card Module – 10лв

9.2 броя бутони – 0.50лв

10.ML144110VP 6 bit DAC – 5лв

11.UA741 Op-Amp – 0.50лв

12.Схема за засичане на нивото на водата – 0.50лв

Общо 87.44лв

Средна цена на кафемашина с подобни функции 150лв

**VII. Бъдещо развитие**  
 В бъдеще смятаме да довършим възпроизвеждането на звук и да запишем голям брой записи.Ще заснемем нов клип, който ще бъде за развлечение и ще дадем кафемашината за реално използване.

**Благодарим за вниманието и съдействието!**