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

ДОКУМЕНТАЦИЯ

Тема: Автоматична лампа

Изготвил:
Ивайло Каньов 10”Б” клас

Ръководител:
Енчо Шахънов

София
2022 г.

УВОД

Във времена, в които ежедневието на средностатистическия човек е натоварено, всяко предимство е добре дошло. Тъй като водим бърз начин на живот, нерядко се случва да забравяме неща – както ключови, така и маловажни. Пример за това са именно лампите – често използвани и в същото време често неизгасяни. Поради невнимание или чиста проява на леност, този навик има своите последствия върху нас и природата. Проблемът се корени именно в енергийната неефективност – пропиляно е голямо количество електроенергия, което може да се използва за други дейности. По този начин се хабят и ресурсите, които са необходими за производството ѝ. В България голям процент от електроенергията се произвежда в ТЕЦ-ове, които използват невъзобновяеми ресурси – въглища. От своя страна при изгарянето им с получената енергия се отделят и въглеродни емисии, които замърсяват въздуха и са причина за парниковия ефект – явление, предизвикващо повишаване на температурата на земната повърхност и долните части на атмосферата. Промяната в климата води до изчезването на много видове растения и животни, което разрушава установения ред в природата. За да се намали разходът на електроенергия, може да се използват умни осветителни тела, които при засичане на движение в тъмна среда, светват за определен период от време.

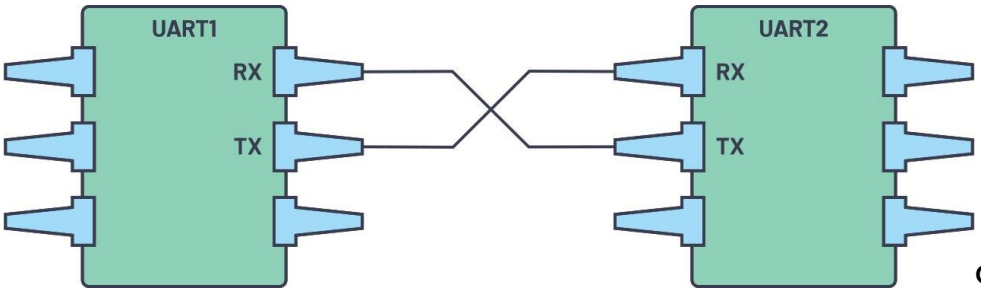
ПЪРВА ГЛАВА

ТЕОРИЯ

1.1 Видове серийни комуникации

а) UART (Universal Asynchronous Receiver/Transmitter) [1], [2]

При UART комуникация се използват само два проводника за предаване на информация. Това се случва асинхронно, което ще рече, че вместо тактов сигнал предаващият чип ще добавя начален и краен бит към пакета с данни, който се прехвърля. Тези битове определят кога приемащият чип да започне да чете битовете. Когато приеме начален бит, приемащият UART ще започне да чете при определена честота, нар. „скорост на предаване“. И приемащото и предаващото устройство трябва да работят с приблизително една и съща скорост на предаване. Предимството на UART е, че информацията се изпраща директно, като няма нужда от изчакване на други устройства (фиг. 1.1а).

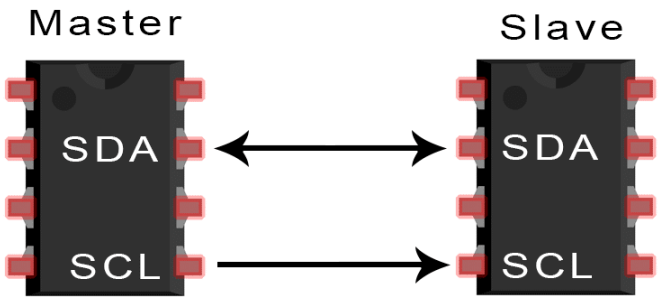


Фиг. 1.1а

б) I2C (Inter-Integrated Circuit) [1], [3]

Тази комуникация е предназначена да позволи на множество "подчинени устройства" да комуникират с едно или повече "главни устройства" (принцип главен-подчинен (master-slave)). Предаването на информация е на база шина, съставена от SDA (Serial Data) и SCL (Serial Clock) - SDA е сигнал за данни, а SCL - синхронизиращ сигнал (часовник), който определя кое устройство кога трябва да изпраща или приема информация.

Предимството на този протокол е, че позволява на микроконтролер да се свърже с множество устройства. Всеки подчинен има уникален 7 или 10 битов адрес, който се използва за достъпването му. Недостатъците са, че I2C е бавна серийна комуникация и че по този начин само едно устройство комуникира с master в даден момент (фиг. 1.1б).



Фиг. 1.1б

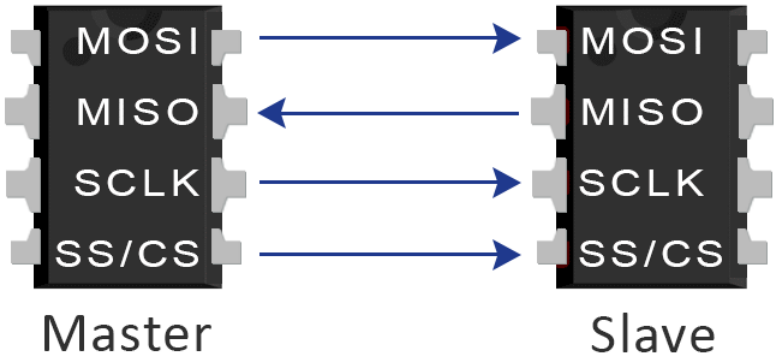
в) SPI (Serial Peripheral Interface) [1], [4]

SPI е друг популярен сериен протокол, използван за по-бързи скорости на данни от около 20Mbps. Той използва общо четири проводника, а именно SCK (серийна тактова линия), MISO (Master Out Slave In), MOSI (Master In Slave Out) и SS/CS (Chip Select). За разлика от UART, SPI използва формат master-to-slave за управление на множество подчинени устройства само с един главен.

MISO и MOSI действат като Tx и Rx на UART, използвани за предаване и получаване на данни. Избор на чип се използва за избор с кое подчинено устройство главният иска да комуникира.

Тъй като SPI е синхронен протокол, той използва вграден часовник от главния, за да гарантира, че и главното, и подчинените устройства работят на една и съща честота. Това означава, че двете устройства вече не трябва да договарят скорост на предаване.

Със своята способност за свързване към множество подчинени устройства, пълна дуплексна комуникация и по-ниска консумация на енергия от другите синхронни протоколи като I2C, SPI се използват в устройства с памет, цифрови карти с памет и други (фиг. 1.1в).



Фиг. 1.1в

1.2 Безжични технологии

a) Bluetooth [5]

Bluetooth представлява технология за комуникация между две или повече устройства без нужда от намесата на потребителя и **с потреблението на съвсем малко енергия**. Тя предава данни, използвайки радио вълни с ниска мощност. Комуникацията се осъществява на честота от 2.45 GHz. Сигналите, които Bluetooth излъчва са много слаби – само около един миливат. Създадена с цел направата на PAN (Personal Area Network), Bluetooth е подходяща за връзката на две устройства на къси разстояния (не повече от десет метра в радиус) и предаването на малко количество данни.

б) WiFi [6]

WiFi е популярна технология, която позволява електронни устройства да обменят данни или да се свързват с Интернет безжично, използвайки радиовълни. Това е технология на безжичната мрежа (WLAN) базирана на спецификациите от серията IEEE 802.11. Работеща на 2, 4 и 5 GHz, има възможност за предаване на информация съответно с висока скорост на близки разстояния или с по-ниска скорост на по-далечни.

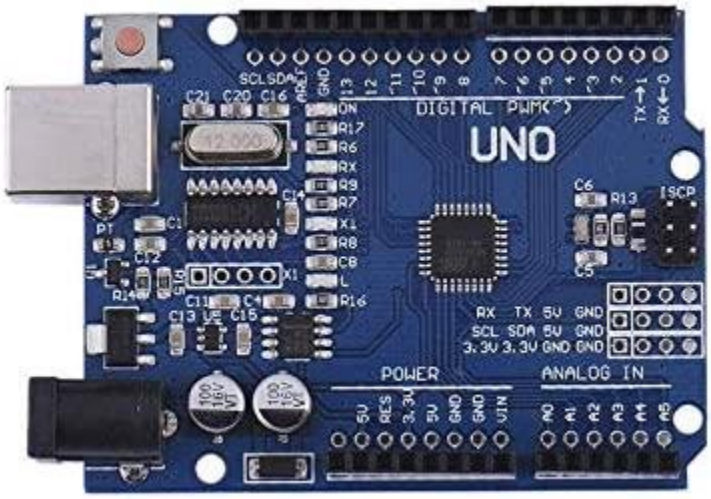
в) LTE [7]

В телекомуникациите 4G е стандарт за висикоскоростно предаване на информация за мобилни устройства, базирана на 2G и 2,75G технологии. LTE предлага по-голяма скорост на връзката - значително увеличени са скоростите на качване и изтегляне, както и ниска латентност на прехвърлянето на данни. Предимство пред WiFi и Bluetooth е, че има много малко време закъснение в мрежата (<5ms). Важен недостатък е, че трябва да се поддържа постоянна свързаност между устройствата, което директно води до висока консумация.

1.3 Избор на хардуерна платформа

а) Избор на основна платка [8]

Предвид наличните познания и платки проектът бе разработен с Arduino Uno (фиг. 1.3а), тъй като този микроконтролер предлага най-пълна функционалност за най-малка цена. Платката разполага с 14 цифрови входно-изходни порта, 6 от които могат да са PWM (ШИМ) изходи, 6 аналогови входа и кварцов резонатор на 16MHz. Захранването на платката се осъществява през USB порта на всеки компютър или от външен захранващ източник, като батерия или мрежов адаптер с напрежение 7 ÷ 12V DC. Arduino Uno поддържа UART, I2C и SPI серийни протоколи.

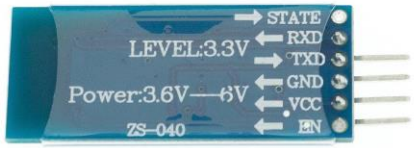


Фиг. 1.3а

б) Избор на Bluetooth модул [9]

Използван е модул, изграден чрез сериен Bluetooth HC-06 модул на базата на чип (BC352). Изведен е конектор (фиг. 1.3б) с четири извода (+5V, GND, TX, RX), като първите два са за захранване на модула, а другите два са за сериен интерфейс за връзка (фиг. 1.3в). Платката разполага с вградена антена и светодиод, който служи като индикатор за осъществена връзка, също така и за включено захранване. Скоростта на обмен (Baud rate) е 9600 bps, което означава, че UART комуникация между нея и основната платка е възможна.

Фиг. 1.3б



Фиг. 1.3в

в) Избор на PIR – сензор за движение [10]

Модулът представлява сензор, който реагира на обекти – източници на инфрачервена светлина от 7μm до 14μm. Подходящ е за засичането на хора и животни, тъй като те отделят топлина и излъчват в инфрачервения спектър, както и е значително евтин. Сензорът за засичане на движение (фиг. 1.3г) може да се използва в охранителни системи за засичане на нарушители, за включване на лампи и др. Има постоянна консумация под 1mA, ъгъл на засичане от 120°, като може да се променя чувствителността (от 3 до 7 метра) и времето на задействане (0,3сек. / 10мин.). Захранващото напрежение е +5V ÷ +14V DC.

Фиг. 1.3г

г) Избор на фоторезисторен датчик

Модулът е изграден с фоторезистор (фиг. 1.3д). Предимствата му са ниската цена и наличието на аналогов и цифров изход, чийто изходен сигнал постъпва под формата на напрежение или логически нива. Чрез тример-потенциометър може да се регулира чувствителността на фоторезистора. Модулът е снабден с два индикаторни светодиода, съответно за включено захранване и за изходното ниво на цифровия изход. Захранващото напрежение е 3,3V ÷ 5V DC.

Фиг. 1.3д

д) Избор на релеен модул [11]

Модулът съдържа едно реле, което се управлява с логически нива. При подавано входно ниво лог. нула на вход (IN), релето е в активен режим, с което светва зелен светодиод. Респективно при входно ниво лог. единица, релето се деактивира и зеленият светодиод изгасва. Контактните изводи на релето са изведени на терминал и са обозначени (NO, COM, NC) (фиг. 1.3е). Като индикатор за включено захранване служи червеният светодиод. Захранващото напрежение е +5V DC (фиг. 1.3ж).

Фиг. 1.3е

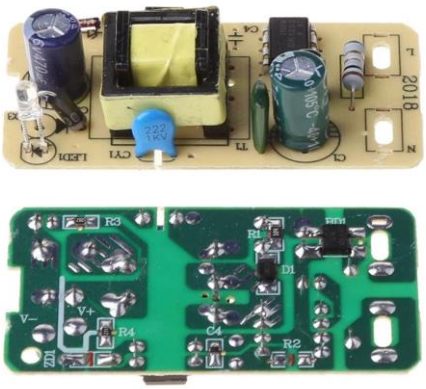
Фиг. 1.3ж

е) Избор на захранване

Използван е AC/DC адаптер платка (фиг. 1.3з) със следните характеристики:

- Входно напрежение - AC 100V - 240V с честота 50 - 60Hz;
- Изходно напрежение – DC 12V 1A;

Адаптерът има защита от късо съединение на изхода, претоварване и прегряване. Той се използва, за да захрани Arduino-то.

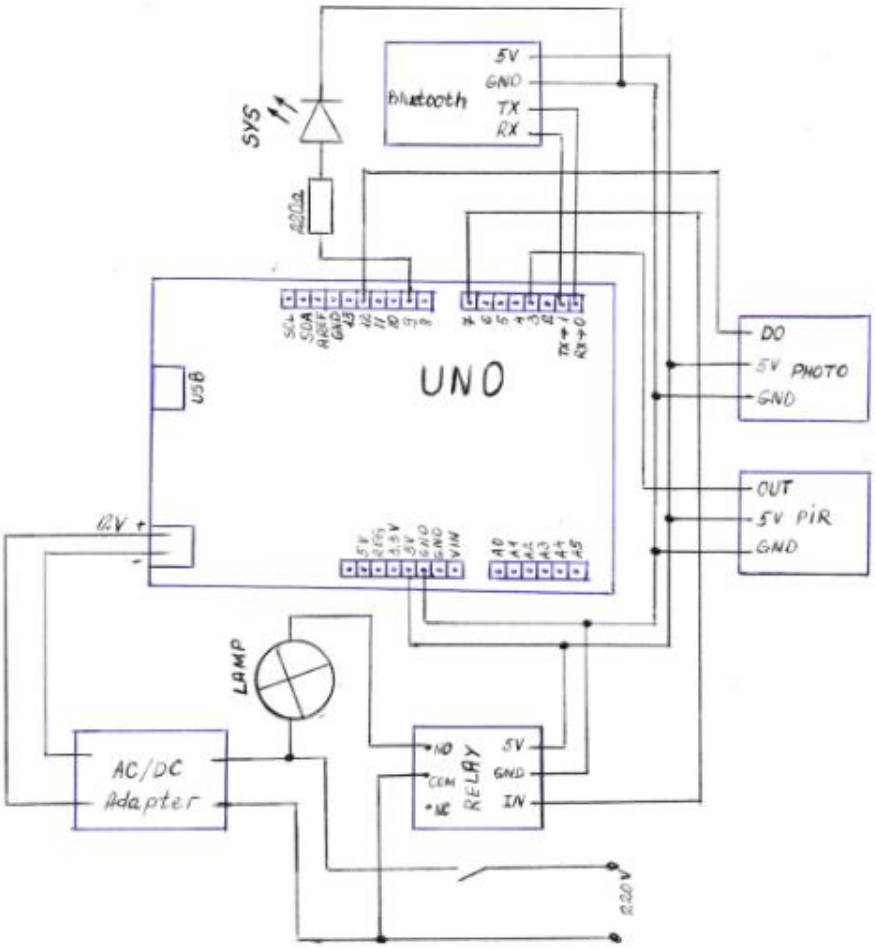


Фиг. 1.3з

ВТОРА ГЛАВА

ПРАКТИКА

2.1 Схема на свързване



Фиг. 2.2а

На фигура 2.2а е показана блоковата схема на устройството.

- 5V и GND на Arduino-то се подават на съответните пинове на всички сензори, релеен и Bluetooth модул;
- Управляващият вход на релейния модул е свързан към 7-ми пин;
- Изходът DO на фоторезисторния датчик е свързан към пин 12;
- Изходът на PIR сензора е свързан към пин 3;
- RX-ът и TX-ът на Bluetooth модула са свързани съответно с 1-ви (TX) и 0-ев (RX) пин;
- Анодът на системния LED е свързан към 9-ти пин, като в тази верига е включен и 220Ω резистор за ограничаване на тока през светодиода;
- За прекъсване на веригата на лампата използваме нормално отворения контакт на релето (NO) ;
- AC/DC адаптерът се захранва с 220V, а неговият изход, който е 12V, се подава за захранване на основната платка;

2.2 Основно описание на софтуера

В началото на кода се дефинират всички пинове, на които са свързани отделните сензори, диоди и реле. На дигитален пин 13 се намира LED, който е на самото Arduino Uno, на 12-ти - фоторезисторен датчик, на 9-ти – LED, който показва дали системата е включена, на 7-ми – релеен модул и на 3-ти – PIR. Важно е да се отбележи, че за UART комуникацията се използват нулевият и първият дигитален пин като съответно RX и TX (към тях е свързан кръстосано Bluetooth модулът) (фиг. 2.2б) **[12]**.

```
1   #define LED_PIN_1 13
2   #define PHS_PIN 12
3   #define LED_PIN_2 9
4   #define RELAY_PIN 7
5   #define PIR_MS_PIN 3
```

Фиг. 2.2б

От 8-ми до 11-ти ред се инициализират глобалните променливи, които се използват, за да се контролира системата (фиг. 2.2в). Стрингът input_string се използва като буфер, в който се запазва въведената команда от потребителя, подадена от Bluetooth терминал.

```
8   int pirMsCalibrationTime = 30; // в секунди
9   boolean sys = true; // дали системата е включена
10  boolean light = false; // дали лампата свети
11  int lightTime = 40;  // в секунди; колко време ще свети лампата при необходимите условия
12
13  String input_string = "";
```

Фиг. 2.2в

Функцията setup() се извиква веднъж в началото на програмата (фиг. 2.2в). В нея се стартира серийната комуникация (17-ти ред) със скоростта на обмен (Baud rate) 9600 bps. Задават се input (12, 3) и output (13, 9, 7) пиновете, като освен това се подава високо напрежение на релейния модул, за да не се задейства и на системния LED – за да индикира, че системата работи. Последно се извиква функцията pir_calibration(), която общо изчаква 35 секунди, за което време сензорът за движение се ориентира в околната среда (фиг. 2.2г).

```
16  void setup() {
17    Serial.begin(9600);
18
19    pinMode(PHS_PIN, INPUT); // фоторезистор
20    pinMode(PIR_MS_PIN, INPUT); // сензор за движение
21
22    pinMode(LED_PIN_1, OUTPUT); // LED за движение
23    pinMode(LED_PIN_2, OUTPUT); // LED за системата
24    pinMode(RELAY_PIN, OUTPUT); // Реле - то пуска или спира тока; работи като ключ в схемата
25
26    digitalWrite(RELAY_PIN, HIGH); // държи релето вдигнато
27    digitalWrite(PIR_MS_PIN, LOW); // първоначално датчикът за движение не засича движение
28    digitalWrite(LED_PIN_1, LOW); // светодиодът свети, когато е засечено движение
29    digitalWrite(LED_PIN_2, HIGH); // светодиодът свети, с което показва, че системата е активна
30
31    pir_calibration(); // сензорът за движение се нуждае от време, за да се ориентира в околната среда
32  }
```

Фиг. 2.2в

```
60  void pir_calibration() {
61    delay(5000);
62
63    Serial.print("Calibrating Motion Sensor ");
64
65    for (int i = 1; i <= pirMsCalibrationTime; i++) {
66      if (i % 5 == 0) Serial.print("_");
67      else Serial.print(".");
68      delay(1000);
69    }
70
71    Serial.println("");
72    Serial.println("DONE!");
73    delay(50);
74  }
```

Фиг. 2.2г

В главната функция loop() се намира логиката на системата (фиг. 2.2д). За да се светне лампата автоматично, трябва да са изпълнени следните условия – системата трябва да бъде включена, PIR **[13]** трябва да е засякъл движение и околната среда трябва да бъде тъмна. Ако всичко това е спазено, светва LED-ът на самата платка, с което се индикира, че системата е задействана. Следващата стъпка е да се светне крушката. Това се случва благодарение на релето, което бива спуснато и съответно ток протича. Подновява се статусът на лампата и започва цикъл, който продължава или докато зададеното време изтече, или в случай, че системата е изключена. След него релето отново се вдига, ток спира да тече към лампата, тя изгасва и индикиращият LED спира да свети.

```
void loop() {
   if (digitalRead(PIR_MS_PIN) == HIGH && digitalRead(PHS_PIN) == HIGH && sys == true) {  // ако е засечено движение + е тъмно + системата е включена
      Serial.println("Motion detected!");

      digitalWrite(LED_PIN_1, HIGH); // светва диода

      digitalWrite(RELAY_PIN, LOW); // спуска релето и лампата светва (затваря се ключът на ел. схема)
      light = true;

      for (int i = 1; i <= lightTime; i++) {
        input_string = Serial.readString();
        commands();
        if (sys != true) break;
        delay(1000);
      }

      digitalWrite(RELAY_PIN, HIGH); // вдига се релето и лампата изгасва
      digitalWrite(LED_PIN_1, LOW); // диодът изгасва
      light = false;
      delay(100);
    }

    input_string = Serial.readString();
    commands();
}
```

Фиг. 2.2д

За да се управлява самата система, се използват команди. Те се въвеждат в Bluetooth терминал, получават се от Bluetooth модула и след това се записват в буфер. Този буфер се разглежда във функцията commands() (фиг. 2.2е). Ако input_string е празен, не се извикват функциите, които се опитват да разпознаят даден вид команда.

```
void commands() {
    if (input_string != "") sysOnOff();

    if (input_string != "") ltChange();

    if (input_string != "") lampMode();

    if (input_string != "") sysInfo();

    if (input_string != "") help();
}
```

Фиг. 2.2е

Първата функция е sysOnOff() (фиг. 2.2ж). Тя проверява дали буферът започва със “sys on” или “sys off”. Ако това е така, съответно в първия случай се включва системата, а във втория – се изключва. Следват актуализиране на статуса на системата, изключване на лампата, ако тя работи и се изпразва буферът с цел да не се влиза в другите функции (това се случва и при тях).

```
88  void sysOnOff() {
89    if (input_string.startsWith("sys on") == true) {
90       if (sys != true) {
91         sys = true;
92         if (light == true) {
93           digitalWrite(RELAY_PIN, HIGH);
94           light = false;
95         }
96         digitalWrite(LED_PIN_2, HIGH);
97         Serial.println("System ON");
98       }else Serial.println("System Already ON");
99       input_string = "";
100
101     }else if (input_string.startsWith("sys off") == true) {
102       if (sys != false) {
103         sys = false;
104         if (light == true) {
105           digitalWrite(RELAY_PIN, HIGH);
106           light = false;
107         }
108         digitalWrite(LED_PIN_2, LOW);
109         Serial.println("System OFF");
110       }else Serial.println("System Already OFF");
111       input_string = "";
112     }
113 }
```

Фиг. 2.2ж

Втората функция ltChange() отговаря за времето, което определя колко дълго ще свети лампата при необходимите условия. Тя проверява дали буферът започва със “sys new lt ”. Ако това е така, този стринг се премахва от input_string заедно с всички празни пространства **[14]**. Започва проверка на подадената стойност (фиг. 2.2з). Ако тя:

- е “default” или “d”, времето за светене ще стане 40 секунди;
- завършва на “s”, означава, че подаденото време ще се разглежда като секунди. Маха се мерната единица, cast-ва се от String към int **[15]** и се проверява дали времето е в рамките на допустимите стойности – от 20 секунди до 15 минути включително. В случай, че всичко е успешно, стойността се запазва. (1)
- завършва на “m”, означава, че подаденото време ще се разглежда като минути. Останалото е аналогично на (1), но с единствената разлика, че при самото записване стойността се умножава по 60, за да се запази във вид на секунди (фиг. 2.2и).

```
115  void ltChange() {
116    if (input_string.startsWith("sys new lt ") == true) {
117        input_string.remove(0, 11);  // removes "sys new lt "
118        input_string.replace(" ", "");  // removes whitespaces
119
120        if (input_string == "default" || input_string == "d") {
121          if (lightTime != 40) {
122            lightTime = 40;
123            Serial.print("Done! New Light Time: ");
124            Serial.print(lightTime);
125            Serial.println(" seconds");
126          }else Serial.println("Light Time Is Already Set To 40 seconds!");
127
128        }else if (input_string.endsWith("s") == true) {
129          input_string.remove(input_string.length()-1);  // removes "s"
130          int seconds = abs(input_string.toInt());  // if input_string is not a valid number => returns 0
131          if (seconds != 0 && seconds >= 20 && seconds <= 900) {  // граници -> от 20 секунди до 15 минути включително
132            lightTime = seconds;
133            Serial.print("Done! New Light Time: ");
134            Serial.print(lightTime);
135            Serial.println(" seconds");
136          }else Serial.println("Incorrect input! Check if the value is between 20 and 900 seconds");
```

Фиг. 2.2з

```
137
138        }else if(input_string.endsWith("m") == true) {
139          input_string.remove(input_string.length()-1);  // removes "m"
140          int seconds = abs(input_string.toInt()) * 60;  // if input_string is not a valid number => returns 0
141          if (seconds != 0 && seconds >= 20 && seconds <= 900) {  // граници -> от 20 секунди до 15 минути включително
142            lightTime = seconds;
143            Serial.print("Done! New Light Time: ");
144            Serial.print(lightTime/60);
145            Serial.println(" minutes");
146          }else Serial.println("Incorrect input! Check if the value is no more than 15 minutes");
147
148        }else Serial.println("Incorrect input! Make sure you follow this pattern: sys new lt [integer][s/m]");
149        input_string = "";
150      }
151  }
```

Фиг. 2.2и

Функцията lampMode() позволява на потребителя да включва или изключва лампата, в случай че системата е изключена (фиг. 2.2к).

```
153  void lampMode() {
154    if (input_string.startsWith("sys lamp on") == true) {
155        if (sys == false && light == false) {
156          digitalWrite(RELAY_PIN, LOW);
157          light = true;
158          Serial.print("Done! Light ON!");
159        }else Serial.println("This feature works when the system is stopped and the lamp does not light!");
160        input_string = "";
161
162      }else if (input_string.startsWith("sys lamp off") == true) {
163        if (sys == false && light == true) {
164          digitalWrite(RELAY_PIN, HIGH);
165          light = false;
166          Serial.print("Done! Light OFF!");
167        }else Serial.println("This feature works when the system is stopped and the lamp lights!");
168        input_string = "";
169      }
170  }
```

Фиг. 2.2к

Функцията sysInfo() предоставя информация за състоянието на системата, лампата и времето за светене (фиг. 2.2л). Тя се съхранява именно в глобалните променливи от фиг. 2.2в.

```
172  void sysInfo() {
173    if (input_string.startsWith("sys info") == true) {
174      Serial.println("INFO");
175
176      Serial.print("-> System: ");
177      if (sys) Serial.println("Active");
178      else Serial.println("Inactive");
179
180      Serial.print("-> Lamp: ");
181      if (light) Serial.println("ON");
182      else Serial.println("OFF");
183
184      Serial.print("-> Light time: ");
185
186      int s = lightTime % 60;
187      int m = ((lightTime - s)/60) % 60;
188
189      Serial.print(m);
190      Serial.print(":");
191      if (s <= 9) Serial.print("0");
192      Serial.print(s);
193      if (m < 2) Serial.println(" minute");
194      else Serial.println(" minutes");
195
196      input_string = "";
197    }
198  }
```

Фиг. 2.2л

Последната функция е именно help(), която принтира наличните команди, с които се управлява автоматичната лампа (фиг. 2.2м).

```
200  void help() {
201    if (input_string.startsWith("sys help") == true || input_string.startsWith("help") == true) {
202      Serial.println("HELP - COMMANDS");
203      Serial.println("sys [on/off] <-> turns the system on or off\n");
204      Serial.println("sys new lt [integer][s/m] <-> sets new light time; [s/m] - seconds or minutes\n");
205      Serial.println("sys lamp [on/off] <-> turns the lamp on or off\n");
206      Serial.println("sys info <-> provides information about the system\n");
207      Serial.println("sys help; help <-> displays this message\n");
208      Serial.println("You don't have to write [ ]");
209      input_string = "";
210    }
211  }
```

Фиг. 2.2м

ЗАКЛЮЧЕНИЕ

Създадена е работеща автоматична лампа, с която може да се осъществи безжична комуникация, използвайки Bluetooth. За настройване на системата се използват команди, въвеждани в Bluetooth терминал. В бъдеще може да бъде добавена постоянна памет, в която да се записват настройките, за да не се налага повторното им въвеждане след рестартиране или изключване на лампата. Освен това може да бъде направено удобно приложение, с което да се улесни работата на потребителя с нея.

ИЗПОЛЗВАНА ЛИТЕРАТУРА

[1], https://bg.denizatm.com/pages/51535-how-uart-spi-and-i2c-serial-communications-work-and-why-w

[2], https://www.electronicevolution.bg/bg-news-details-65.html

[3], https://www.electronicevolution.bg/bg-news-details-46.html

[4], https://www.electronicevolution.bg/bg-news-details-47.html

[5], https://revo.bg/blog/vsichko-koeto-tryabva-da-znaem-za-bluetooth/

[6], https://revo.bg/blog/kakvo-predstavlyava-wi-fi/

[7], https://shopdelta.eu/lte-tehnologiya_l13_aid1049.html

[8], https://elimex.bg/product/71201-kit-k2014-razvoyna-platka-s-atmega328p-smd-usb-b

[9], https://elimex.bg/product/82656-kit-k2189-serienbluetooth-hc-06-modul-s-bc352

[10], https://elimex.bg/product/71202-kit-k2015-pir-%E2%80%93-senzor-za-dvizhenie

[11], https://elimex.bg/product/71200-kit-k2013-modul-s-rele-aktivno-nisko-nivo

[12], https://create.arduino.cc/projecthub/RucksikaaR/interfacing-the-hc-06-bluetooth-module-with-arduino-f9c315

[13], https://create.arduino.cc/projecthub/electropeak/pir-motion-sensor-how-to-use-pirs-w-arduino-raspberry-pi-18d7fa

[14], https://stackoverflow.com/questions/32811197/how-to-remove-white-spaces-from-string-in-arduino

[15], https://forum.arduino.cc/t/extracting-numeric-data-from-a-string-coming-from-a-serial-port/221629/5

СЪДЪРЖАНИЕ