Wnioski z wykonania laboratoriów

Ćwiczenie 3:

FAQ: Układ KY-026 jest bardzo prosty w obsłudze, podłączenia można dokonać przez przedłużki do kabli. Należy uważać n odwrócone dane w Monitorze Szeregowym – 0 określa wykrycie ognia, a 1 do jego brak. Można pobawić się potencjometrem, żeby zwiększyć czułość. Wprowadzanie interwału badania odbywa się z użyciem funkcji delay().
W ćwiczeniu 2 w miejscu XXX powinno być LOW (bo sygnał 0, czyli wykryty ogień – wtedy uruchamia się buzzer). Błąd jest w programie w 10 linii – zamiast HIGH powinno być LOW. Schemat dźwięku buzzera można modyfikować poprzez zmianę delay(XXX) w linii 9 i 11.
PILNOWAĆ, BY KY-026 został poprawnie podłączony, bo mogą usmażyć układ.

Weryfikacja wiedzy: ogólnie ciężko, bo to jedno z pierwszych ćwiczeń
1. Sprawdzić, czy umie zmienić interwał czasowy między pokazywaniem się wyniku w Monitorze Szeregowym delay().
2. Sprawdzić, czy rozumie zasadę działania buzzera (i czemu w „niekompletnym kodzie" w 10 jest błąd).
3. Zapytać, czemu pin 7 jest INPUT a 12 OUTPUT.

Ćwiczenie 4:

FAQ: Z doświadczenia – rezystor w połączeniu na RYS.1 nie musi być koniecznie ten. Jeśli fotorezystor będzie szalał (wyniki w Monitorze Szeregowym albo będą niewspółmierne do oświetlenia, albo jednakowe) to GND z fotorezystora podłączyć do osobnej GND na arduino (najlepiej tego w sekcji POWER), a nie do wspólnego z diodami LED. Żeby działało wszystkie 7 diod to podłączyć od portu 3 do 10 (w ćwiczeniu na rysunku jest bardzo niewyraźnie i widać, jakby było od 2). Kierunek zmieniany poprzez zmianę kolejności podłączeń. Wartości graniczne światła „zgaszonego" o w praktyce 0 (przy dobrym zakryciu fotorezystora).

Weryfikacja wiedzy:
1. Zapytać, czy podłączony odwrotnie fotorezystor zadziała czy nie.
2. Zapytać, czemu podłączamy pod port analogowy, a nie cyfrowy
3. Jak zmienić w 2a podgląd ilości diod na podgląd wartości normalnej?
4. Co się stanie, jak odłączymy co 2 diodę?
5 Jaką wartość maksymalną uda się uzyskać na fotorezystorze w Monitorze Szeregowym?

Ćwiczenie 6:

FAQ: Proponuję do każdego wyprowadzenia „kolorowego" dołączyć kabelek odpowiedzialny za ten kolor (wyprowadzenie pierwsze – kabel czerwony itd.). Odłączanie w drugiej części ćwiczenia kabelków będzie powodowało powstawanie nowego koloru bez składowej odłączonego koloru. Żółty można uzyskać odłączając niebieski (vide Rys1). Sygnalizacja trywialna, jeśli wykona się ją na osnowie gotowego programu.

Weryfikacja wiedzy:

1. Czym się różni dioda ze wspólną katodą/anodą (i różnice w podłączeniu).
2. W jaki sposób (operując tylko na hexadecymalnych wartościach) mając wartości RED GREEN i BLUE stworzyć wartość WHITE?
3. Sprawdzić, czy umie sobie znaleźć wartość hexadecymalną dowolnego koloru.
4. W pkt 2b zamiast żółtego zasymulować inny kolor „na sucho", mając do dyspozycji tylko sprawozdanie (bez odłączania kabelków).

Ćwiczenie 8:

FAQ: NIE ZAPOMNIEĆ WGRAĆ BIBLIOTEKI! Przycisk dobrze jest podłączyć na płytce stykowej między dwoma częściami płytki, bo w takim wypadku nie da rady podłączyć go źle (Rys1). Przy DHT11 i rezystorze podciągającym mogą być problemy „magiczne", czyli podłączenie rezystora przed czujnikiem nie zadziała, ale podłączenie na tej samej ścieżce za czujnikiem już działa prawidłowo. Przy podłączaniu DHT11 dobrze zostawić przestrzeń przed nim wolną (lub nawet podłączyć kabelkami przedłużającymi), żeby nie spalić ogniem zapalniczki czegokolwiek podczas testowania jego działania.

Weryfikacja wiedzy:

1. Zmiana komunikatu w Monitorze Szeregowym (np. żeby pokazywało „Moja aktualna temperatura – X st C").
2. Symulowanie działania TactSwitch bez przycisku (można kabelki zwierać/rozwierać).
3. Zapytać, do czego służy 3 wyprowadzenie w DHT11 (puste, bez roli).
4. Co się stanie, jak w 3.1 w linii 13 zamiast „==" damy „=".
5. Wyrzucić bibliotekę DHT11 z ArduinoIDE i kazać znaleźć przyczynę błędu i go naprawić.

Ćwiczenie 10

FAQ: NIE ZAPOMNIEĆ WGRAĆ BIBLIOTEKI! Układ bardzo prosty w podłączeniu, jednak program może sprawiać wiele problemów – stąd pozostawione komentarze w programie. Kod do zamka to 11111 (linia 48 i 49). Void warning bad() to funkcja użytkownika opisana w linii 21-25 ten zabieg służy temu, by kod był bardziej przejrzysty (w loop() mamy tylko odwołanie do funkcji, nie zaś całe 5 linijek kodu).

Int mkey[] tworzy wektor z 5 miejscami na wpisany przez użytkownika kod. Na tej samej zasadzie można by zrealizować np. przechowywanie hasła. Zmienna l to „licznik" wpisanych przez użytkownika cyfr, potrzebny do stworzenia pętli w linii 40. Ulega on inkrementacji (i++) za każdym przejściem pętli i wyzerowaniu po podaniu komunikatu o dobrym/złym pinie. Dyrektywa voliate dotyczy czysto technicznych spraw, więc jej opis będzie raczej copy/paste z Wikipedii.
W części c, kolejno zadania:

- przenieść treść void warning_bad() w miejsce warning_bad w linii 56
- skopiować cały program i zapisać jako nowa funkcja użytkownika (void warning_bad), po czym w loop() po prostu wywołać tę funkcję
- proste zadanie z dodaniem 2 diod
- zmiana pinu w programie w linii 48 i 49

Weryfikacja wiedzy:

1. Jakie zmiany trzeba by było wprowadzić przy klawiaturze 5x5 w liniach 1-20?
2. Jakie zmiany, by przy wciśnięciu na klawiaturze A program pisał jego/jej imię?