Wyższa Szkoła Handlowa w Radomiu

Systemy Wbudowane

Laboratorium 4

1 Cel zajęć

Niniejszy materiał ma na celu zaprezentowanie możliwości Arduino Leonardo pod kątem tworzenia gotowych aplikacji sprzętowo-programowych, takich jak np. stacja pogodowa,sterownik podczerwieni czy włącznik czasowy.

2 Teoria

Układy Arduino pierwotnie miały za zadanie stanowić wprowadzenie do świata elektroniki dla osób, które dotychczas interesowały się tym zagadnieniem jednak dostępne układy i ich dokumentacja/oprogramowanie były zbyt skomplikowane. Poprzez swoją prostotę, bogatą dokumentację oraz otwartość stało się jednak czymś więcej; coraz większa grupa profesjonalistów zaczęła wybierać płytki Arduino jako bazy dla swoich nowych układów – sterowanie zdalnych robotów, stacji pogodowych, elementów sterujących bądź sygnalizacyjnych. Ponieważ układy te, pomimo prostoty łączenia poszczególnych komponentów i oprogramowania ich, posiadają w pełni funkcjonalne komponenty (takie, jakie stosuje się w droższych/profesjonalnych układach), których działanie nie jest niczym ograniczone bądź obarczone narzutem czasowym, a do tego ich cena jest niezwykle niska (nie licząc w pełni legalnych klonów płytek, które bywają jeszcze tańsze). To z kolei pozwala przypuszczać, że na rynku może pojawić się coraz więcej elektronicznych rozwiązań, których bazą będą właśnie oryginalne Arduino bądź układy tworzone na bazie oryginalnych rozwiązań (lecz w oparciu o ten sam mikroporcesor/płytkę drukowaną).

2.1 Zastosowanie rejestru przesuwnego

Dzięki temu dodatkowemu układowi możemy zarządzać większą liczbą wyprowadzeń przy wykorzystaniu trzech wyprowadzeń z naszego układu.

Samo działanie rejestru jest niezwykle proste. Poprzez pojedynczą linię danych wprowadzamy wartość jednego bitu. Rejestr zapamiętuje te dane w swojej wewnętrzne pamięci (która de facto jest rejestrem – stąd nazwa). Jednak zasada zapisu jest taka, że w pierwszej kolejności rejestr zapisuje bit do najstarszej swojej części. Przy podaniu kolejnej informacji zapisany byt przesuwany jest w stronę młodszej części, natomiast nowy stan zapisywany jest w najstarszej części. Sposób ten determinuje nazwę układu.

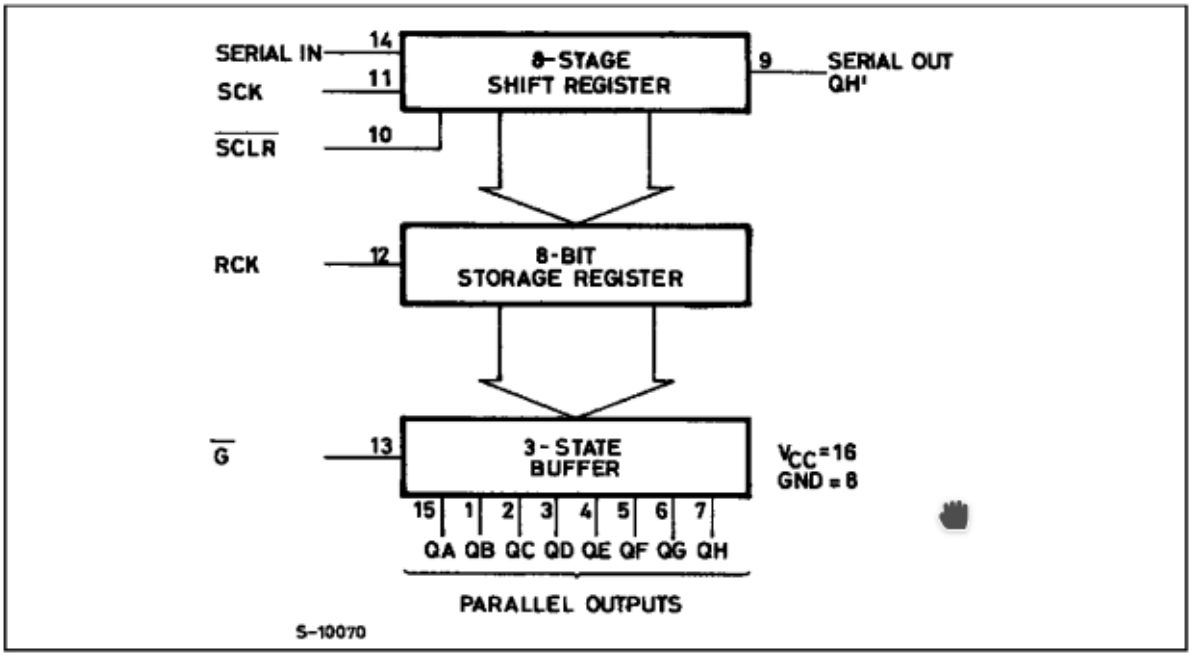
Aby zapis przebiegał płynnie (i nie występowały chwilowe przekłamania w stanach na wyjściu rejestru) wykorzystywane jest wyprowadzenie latch (zatrzaśnięcie), nazywane zegarem rejestru. Jedynie przy zboczu narastającym dane w rejestrze są zapisywane. Ponadto układ posiada drugie wyprowadzenie synchronizujące wprowadzanie danych do układu (które działa analogicznie). Rozwiązanie takie pozwala na bezkolizyjną zmianę wyjść układu na nowe stany logiczne dopiero po zapisie nowych wartości do rejestru. Poniżej przedstawiono tabelę prawdy dla rejestru oraz schemat wewnętrzny:

TRUTH TABLE

INPUTS					OUTPUT
SI	SCK	$\overline{SCLR}$	RCK	$\overline{G}$	
X	X	X	X	H	QA THRU QH OUTPUTS DISABLE
X	X	X	X	L	QA THRU QH OUTPUTS ENABLE
X	X	L	X	X	SHIFT REGISTER IS CLEARED
L	⎍	H	X	X	FIRST STAGE OF S.R. BECOMES "L" OTHER STAGES STORE THE DATA OF PREVIOUS STAGE, RESPECTIVELY
H	⎍	H	X	X	FIRST STAGE OF S.R. BECOMES "H" OTHER STAGES STORE THE DATA OF PREVIOUS STAGE, RESPECTIVELY
X	⎎	H	X	X	STATE OF S.R IS NOT CHANGED
X	X	X	⎍	X	S.R. DATA IS STORED INTO STORAGE REGISTER
X	X	X	⎎	X	STORAGE REGISTER STATE IS NOT CHANGED

X: DON'T CARE

LOGIC DIAGRAM



2.1.1 Przykładowa implementacja rejestru przesuwnego

Klasycznym przykładem działania opisywanego rejestru jest podłączenie do niego 8 diod świecących i zapalanie ich bądź wygaszanie. Do naszych celów wykorzystamy także wyświetlacz oraz dwa przycisk. Przykładowo wyświetlacz podłączamy jak w przykładach w instrukcji 4 (wyprowadzenie 2,3,4,5 oraz 11,12). Rejestr przesuwny pod złącza 8,9 i 13. Z kolei przyciski podłączymy pod pin 6 i 7 (wykorzystamy tym samym niemal wszystkie wyjścia). Jeżeli chodzi o budowę samego układu należy wykorzystać wiedzę z zajęć poprzednich oraz przykładowe rozwiązania dostępne w odnośnikach w sekcji materiały.

Zasada działania naszego rozwiązania jest prosta – jeden z przycisków powoduje zwiększanie, drugi zmniejszanie wartości liczbowej o jeden. Wartość powinna zostać wyświetlona na wyświetlaczu, z kolei rejestr powinien odpowiednio rozświetlić odpowiednie diody.

Kod dla powyższego przykładu:

```
1  #include <LiquidCrystal.h>
2
```

```
3  #define SCLCK 8
4  #define LPIN 13
5  #define SDPIN 9
6
7  #define NUMUP 6
8  #define NUMDOWN 7
9
10 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
11
12 byte val = 0;
13
14 void showShiftValue(byte val);
15
16 void setup()
17 {
18     pinMode(SCLCK, OUTPUT);
19     pinMode(LPIN, OUTPUT);
20     pinMode(SDPIN, OUTPUT);
21     pinMode(NUMUP, INPUT);
22     pinMode(NUMDOWN, INPUT);
23     lcd.begin(16,2);
24     lcd.clear();
25     lcd.print("Wartosc:");
26     lcd.setCursor(0,1);
27 }
28
29 void loop()
30 {
31     delay(700);
32     if (digitalRead(NUMUP) == HIGH)
33     showShiftValue(++val);
34     if (digitalRead(NUMDOWN) == HIGH)
35     showShiftValue(--val);
36 }
37
38 void showShiftValue(byte val)
39 {
40     lcd.setCursor(0,1);
41     lcd.print("                ");
42     lcd.setCursor(0,1);
43     lcd.print(val);
44     for (byte i = 7; i <= 0; i--)
45     {
46         digitalWrite(LPIN, LOW);
47         digitalWrite(SCLCK, LOW);
48         if (val & (1<<i))
49         digitalWrite(SDPIN, HIGH);
50         else
51         digitalWrite(SDPIN, LOW);
52         digitalWrite(SCLCK, HIGH);
53         digitalWrite(SDPIN, LOW);
54         digitalWrite(LPIN, HIGH);
55         delay(100);
56     }
57     digitalWrite(SCLCK, LOW);
58     digitalWrite(LPIN, HIGH);
59 }
```

W przykładzie kodu najważniejszą częścią jest ręcznie napisana funkcja przesyłania bitów do rejestru przesuwnego. Rozwiązanie to spowodowane jest słabą wydajnością rejestr (nie może on jednocześnie podać napięcia

większego niż 70 mA na wszystkie wyprowadzenia). Dzięki ręcznemu podawaniu wartości każdego bitu i odczekiwaniu pomiędzy kolejnymi przesłaniami ok 100 ms (+ czas na wykonanie kolejnych rozkazów) każdy LED będzie zapalał się indywidualnie lecz stabilnie.

2.2 Czujnik temperatury

Jeden z powszechniej stosowanych komponentów układów wbudowanych. Obecnie niemal każdy układ wbudowany posiada przynajmniej jeden czujnik temperatury badający np. temperaturę własną układu. Często zdarza się, że czujniki wbudowywane są by badać temperaturę otoczenia, temperaturę cieczy, innego układu wbudowanego itp. Dołączony do zestawu startowego czujnik jest jednym z najpopularniejszych. Posiada interfejs 1-wire charakteryzujący się przesyłaniem informacji po jednej żyle. Sam kabel powinien posiadać 3 żyły – dodatkową dla zasilania i masy (chociaż może też działać w trybie pasywnym – wtedy zasilanie realizowane jest po linii danych). Rozdzielczość wyniku może być ręcznie ustawiana między 9 a 12 bitami. Dokładność pomiaru to 0.5 stopnia Celsujsza. Układ posiada pamięć wewnętrzną typu EEPROM pozwalającą na zapis temperatury porównawczej (niska/wysoka), a przekroczenie zadanych wartości powodować będzie wysłanie sygnału alarmowego. Rozwiązanie to pozwala oszczędzić czas i miejsce przestrzeni programu na napisanie funkcji alarmowej.

Figure 2. Temperature Register Format

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	**BIT 14**	**BIT 13**	**BIT 12**	**BIT 11**	**BIT 10**	**BIT 9**	**BIT 8**
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

Table 1. Temperature/Data Relationship

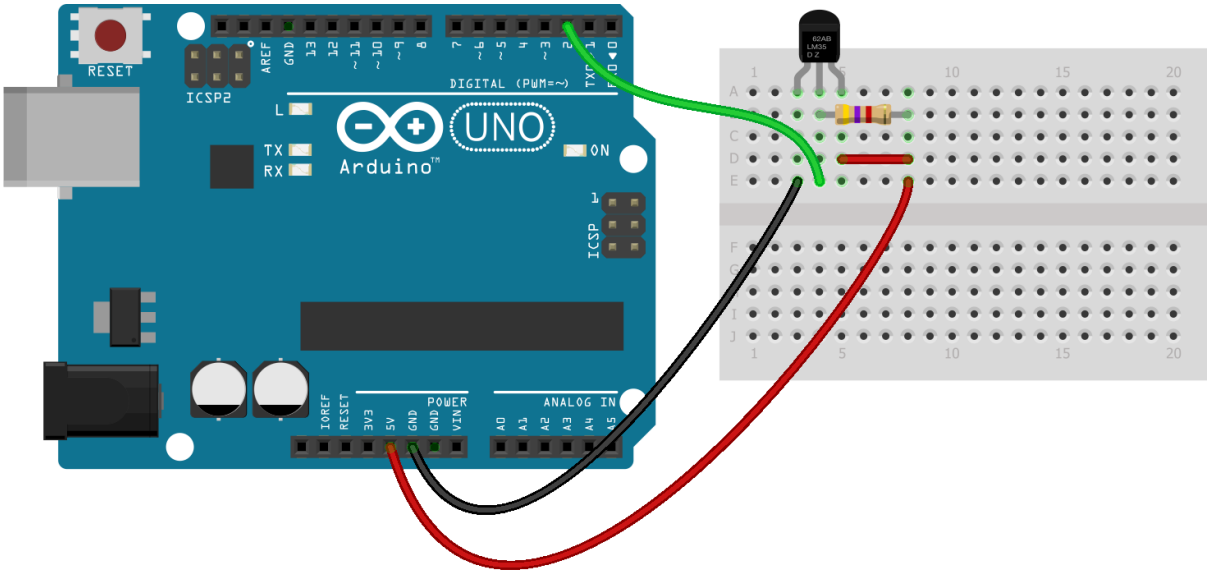
TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

**The power-on reset value of the temperature register is +85°C.*

Powyższa tabela przedstawia układ rejestru odczytu temperatury, gdzie 4 najmłodsze bity stanowią część ułamkową temperatury, 7 bitów to zapis całkowity temperatury, a najstarsze 5 bit to znak. Następna tabela pokazuje relację odczytywanej temperatury i jej reprezentację binarną w rejestrze.

2.2.1 Przykład implementacji termometru

Przykładowe podłączenie termometru z Arduino;




```
#include <OneWire.h>
#include <LiquidCrystal.h>
#include <DS18B20.h>

#define ONEWIRE 2

OneWire ow(ONEWIRE);
DS18B20 temp(&ow);

LiquidCrystal lcd(12, 11, 6, 5, 4, 3);

byte address[8];

void setup()
{
    lcd.begin(16,2);
    lcd.clear();
    lcd.print("Temperatura:");
    lcd.setCursor(0,1);
    ow.reset_search();
    while(ow.search(address))
    {
        if (address[0] != 0x28)
            continue;
        if (OneWire::crc8(address, 7) != address[7])
        {
            break;
        }
    }
    temp.begin();
    temp.request(address);
}

void loop()
{
```

```
36      float temperature = temp.readTemperature(address);
37      lcd.setCursor(0,1);
38      lcd.print("                ");
39      lcd.setCursor(0,1);
40      lcd.print(temperature);
41      lcd.print(" st. C");
42      delay(1000);
43  }
```

Powyższy kod wymaga dodania bibliotek OneWire oraz DS18B20. Pierwszą z nich można bez problemu dodać przez mechanizm bibliotek Arduino IDE. Druga należy pobrać (odnośnik [9]). Ponieważ magistrala OneWire pozwala podłączyć wiele urządzeń w ramach jednej linii pierwszym elementem programu jest zdobycie unikatowego, 64 bitowego adresu urządzenia (po którym następuje komunikacja z urządzeniem). W naszym wypadku mamy tylko jedno urządzenie na magistrali (czujnik temperatury). Dlatego tablica adresowa jest jednowymiarowa (będzie zawierać kolejne części adresu podłączonego urządzenia). Kiedy posiadamy już adres program uruchamia odczyt ze wskazanego urządzenia. W pętli głównej odczyt realizowany jest do 1 sekundę.

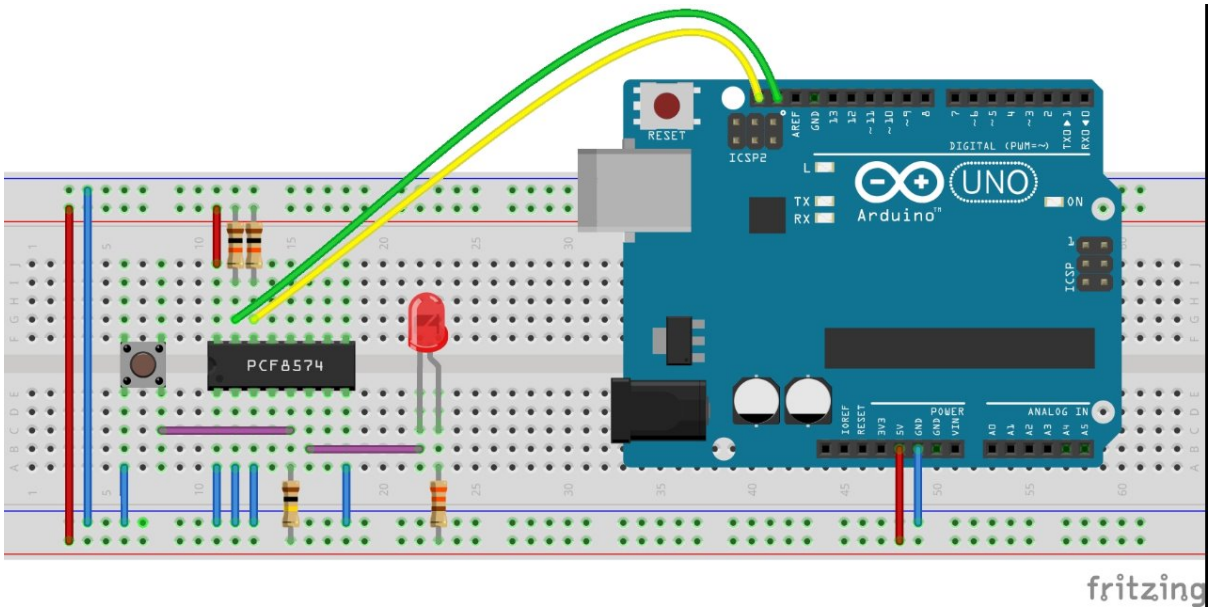
2.3 Rozszerzenie wyjść (Expander)

Arduino Leonardo posiada ograniczoną ilość wyprowadzeń cyfrowych. W niektórych rozwiązaniach wyjść może zdecydowanie brakować. Częściowo problem ten można wyeliminować poprzez wykorzystanie układu rozszerzającego. Z pozoru działanie tego układu może wydawać się podobne do działania rejestru przesuwnego. O ile jednak rejestr potrafił przechowywać i emitować zapisane w nim wartości, o tyle expander pozwala na zapis i odczyt danych z poszczególnych wyprowadzeń. Ponadto umożliwia obsługę przerwań.

Posiadany w zestawie układ ma 8 dodatkowych wyprowadzeń. Obsługa zajmuje 2 wyprowadzenia na płytce Adruino. W związku z tym zyskujemy 6 dodatkowych wyprowadzeń w naszym układzie. W przypadku podłączenia większej ilości elementów zysk jest znacznie większy – expander działa w oparciu o magistralę I2C. Oznacza to, że kolejne układy mogą być podłączane w ramach tych samych wyprowadzeń.

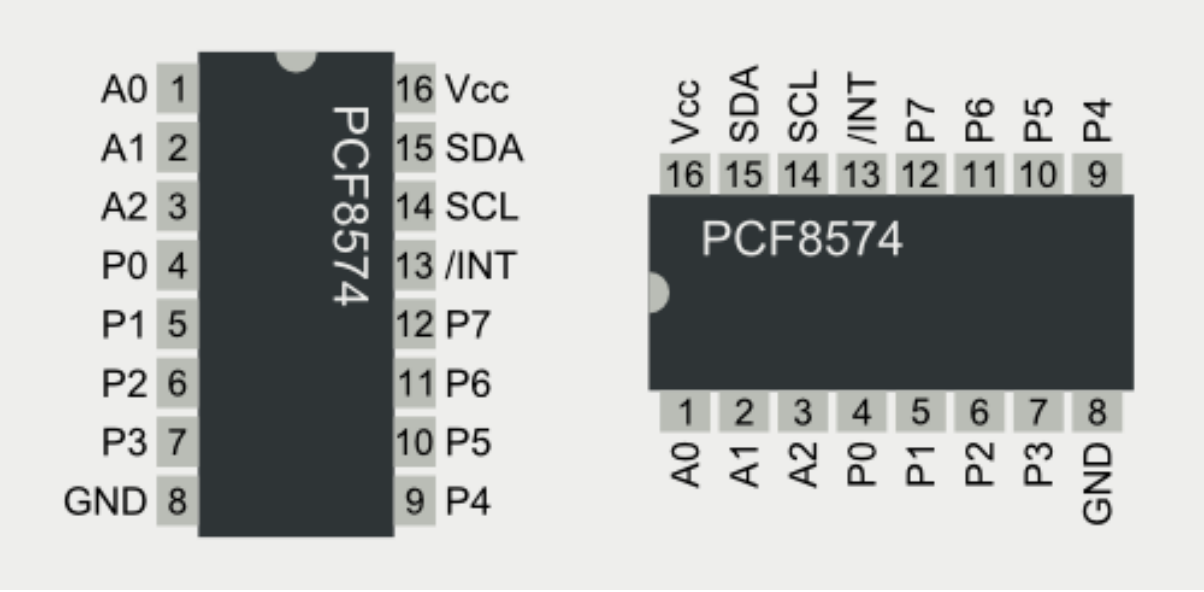
UWAGA! Arduino Leonardo posiada NIEZALEŻNE wyprowadzenia magistrali I2C (wykorzystywana do komunikacji z rozszerzeniem). W związku z tym nie tracimy żadnego wyprowadzenia cyfrowego czy analogowego.

Układ pozwala na ustawianie kolejnych ekspanderów wedle określonego szyku (kaskadowo) poprzez 3 linie adresowe (które, jeżeli nie będą przez nas wykorzystane MUSZĄ być podłączone do masy – inaczej układ nie będzie chciał działać). Najprostszym przykładem pokazującym działanie ekspandera będzie powtórzenie przykładu z pierwszego ćwiczenia – podłączenie diody oraz przycisku:



Proszę zauważyć, że do wyprowadzenia ekspandera została podłączona katoda, nie anoda. Rozwiązanie to eliminuje problem słabszego świecenia diody (ekspander posiada zbyt niskie natężenie prądu). W powyższym przykładzie zastosowano nastepujące oporniki: 33 ohm dla połączenia +5V dla diody, 10 kohm przy połączeniu do I2C (dodatkowe zasilanie) oraz 100 kohm dla przy podłączeniu zasilania do przełącznika z wyprowadzeniem PCF8574.

Wygląd układu (+ opis wyprowadzeń):



Z powyższego wynika, że sygnał przycisku podłączony jest do wyprowadzenia P1, natomiast dioda do P2. Jak już zostało wspomniane każde urządzenie na magistrali I2C ma swój numer. W naszym wypadku numer urządzenia będzie wynosił 0x20 (opis ustalenia adresu można znaleźć w materiale [13]). Ewentualnie można posłużyć się kodem na wykrycie adresu:

- A0-A2 – wyjścia adresowe (służą do adresacji układów na magistrali)
- P0-P7 – kolejne wejścia/wyjścia cyfrowe układu. Działanie takie samo jak wyprowadzeń cyfrowych Arduino Leonardo

- GND – masa
- Vcc – zasilanie 5V
- SDA (Serial Data) – linia danych szeregowych
- SCL (Serial Clock Line) – linia zegara taktującego
- /INT – sygnał generowanego przerwania przez wskazany port (sygnał ten jest zanegowany)

```
#include <Wire.h>

void setup()
{
    Wire.begin();
    Serial.begin(9600);
    Serial.println("\nWykrywanie adresu I2C");
}

void loop()
{
    byte error, address;
    int nDevices;
    Serial.println("Rozpoczynam skanowanie...");
    nDevices = 0;
    for(address = 1; address < 127; address++ )
    {
        Wire.beginTransmission(address);
        error = Wire.endTransmission();
        if (error == 0)
        {
            Serial.print("Adres ");
            Serial.print(++nDevices);
            Serial.print(" urz#dzenia I2C: 0x");
            if (address<16)
            Serial.print("0");
            Serial.print(address,HEX);
            Serial.println("  !");
        }
        else if (error==4)
        {
            Serial.print("Nieznay blad na adresie 0x");
            if (address<16)
            Serial.print("0");
            Serial.println(address,HEX);
        }
    }
    if (nDevices == 0)
    Serial.println("Brak urzadzen I2C\n");
    else
    Serial.println("Zakonczono\n");
    delay(5000);
}
```

Powyższy kod to kosmetycznie zmodyfikowana wersja aplikacji z materiału [14].

Przykładowy kod wykorzystania ekspandera:

```
#include <Wire.h>

#define ADDR 0x20
```

```
4  #define SWITCH 1
5  #define LED 2
6
7  void setup()
8  {
9      Wire.begin();
10 }
11
12 void loop()
13 {
14     volatile uint8_t pinState;
15     uint8_t pressed;
16     Wire.requestFrom((uint8_t) ADDR, (uint8_t) 0x01);
17     while(Wire.available() < 1)
18         pinState = Wire.read();
19     if (pinState & (1 << SWITCH))
20         pressed = HIGH;
21     else
22         pressed = LOW;
23     if (pressed)
24         pinState |= (1 << LED);
25     else
26         pinState &= ~(1 << LED);
27
28     Wire.beginTransmission((uint8_t) ADDR);
29     Wire.write(pinState);
30     Wire.endTransmission();
31     delay(500);
32 }
```

W przykładzie została wykorzystana gotowa biblioteka obsługi magistrali I2C. Dzięki niej nie musimy martwić się o wysyłanie danych oraz stan wejścia zegara taktującego (dba o to biblioteka). Na samym początku deklarujemy trzy stałe: adres urządzenia, numer wyprowadzenia przycisku oraz wyprowadzenie diody. W przypadku pętli setup uruchamiamy transmisję po magistrali.

W pętli programu czytamy jeden bajt ze wskazanego urządzenia (jeden bajt gdyż mamy tylko 8 wyprowadzeń). Następnie sprawdzamy stan bitu podłączonego przycisku. W zależności od jego wartości ustawiany jest bit odpowiedzialny za świecenie diody – stan ten jest wpisywany w bajt pobrany z ekspandera. Na koniec bajt ten przesyłany jest z powrotem do rozszerzenia.

2.4 Podczerwień

Podczerwień jest nadal powszechnie wykorzystywanym systemem transmisji danych. Najpopularniejsze są oczywiście piloty telewizyjne i układy sterowania w środowisku z wysokimi zakłóceniami elektromagnetycznymi. Podczerwień wykorzystywana jest też niekiedy w rozwiązaniach światłowodowych (okablowanie wielomodowe w krótkim torze transmisyjnym).

Nośną sygnału jest światło podczerwone. Najczęściej nadajnik i odbiornik muszą być ustawione naprzeciw siebie (w lepszych układach możliwe są pewne odchylenia, sięgające nawet 60 stopni).

Podczerwień jest transmisją jednostronną (tzw. simplex). Nadajnik może jedynie generować sygnały, a odbiornik tylko odbierać. Aby transmisja była obustronna trzeba byłoby zamontować po obu stronach zarówno diodę podczerwieni jak i odbiornik.

Podłączenie diody emitującej nie różni się niczym od podłączania standardowej diody [16]. Podobnie sprawa wygląda z podłączeniem odbiornika [17]. Samo oprogramowanie sprowadza się do wykorzystania dostępnej biblioteki IRremote.h (którą trzeba pobrać poprzez zarządzanie bibliotekami w środowisku Arduino IDE).

2.4.1 Przykład odczytu emisji

Za przykład emisji niech posłuży nam pilot dostępnego w sali projektora. Po zmontowaniu układu i przepisaniu kodu możemy zacząć wciskać poszczególne przyciski by sprawdzić kody poszczególnych komend:

```
1  #include <IRremote.h>
2  #include <IRremoteInt.h>
3
4  #define irPin 11
5  IRrecv irrecv(irPin);
6  decode_results results;
7
8  void setup()
9  {
10     Serial.begin(9600);
11     irrecv.enableIRIn();
12 }
13
14 void loop()
15 {
16     if (irrecv.decode(&results))
17     {
18         Serial.print("0x");
19         Serial.println(results.value, HEX);
20         delay(250);
21         irrecv.resume();
22     }
23 }
```

Wyświetlane numery można zapisać – będą potrzebne by za pomocą diody emitować odpowiednie sygnały do projektora.

2.5 Przykład emisji

Mając już kody dostępne z pilota można napisać aplikację sterującą projektorem. W przesyle do obcego odbiornika problemem może być ustalenie producenta odbiornika (każdy odbiornik charakteryzuje się zmienną ilością danych przesyłanych z różną częstotliwością!). W materiale [18] (będącym odnośnikiem do biblioteki nagłówkowej) można znaleźć wszystkie funkcje przesyłające dane. W naszym przypadku możemy spróbować z przesyłem Sharp. Jeżeli wybór okaże się chybiony zawsze możemy zmienić metodę przesyłającą/wykorzystać funkcję przesyłu surowego (sendRaw), w której dane podajemy w tablicy bitowej.

```
1  #include <IRremote.h>
2  #include <IRremoteInt.h>
3
4  #define IRLOOP 3 //ilosc powtorzen kodu - trrzeba wpisac odpowiednia ilosc
5  #define IRCODE 0x20DFDF20 //nalezy wpisac kod/kody wykryte w poprzedniej
       aplikacji
6
7  IRsend irsend;
8
9  void setup()
10 {
11 }
12
13 void loop()
14 {
15     for (int i = 0; i < IRLOOP; i++)
16     {
```

```
17          irsend.sendSharpRaw(IRCODE, 32);
18      }
19  }
```

Jak można wywnioskować wysyłanie kodów jest żmudniejsze – wymaga bowiem wiedzy w jaki sposób wysyłane są kody; znajomość samych kodów jest również wymagana. Powyższy kod można użyć jedynie w przypadku gdy dioda emitująca jest podłączona pod pin 3 (złącze wykorzystywanego licznika do taktowania emisji sygnału). Jeżeli chcemy wykorzystać inne złącze/inny licznik to należy zmodyfikować kod biblioteki IRremote.

3 Zadania do wykonania

1. Czy istnieje możliwość jednoczesnego BEZPIECZNEGO dla rejestru zapalenia wszystkich LED jednocześnie? Jeżeli tak należy zaimplementować to rozwiązanie
2. Jak działa funkcja shiftOut z biblioteki obsługi rejestru przesuwnego?
3. Pracując w grupie (2 osoby) należy przetestować możliwość połączenia szeregowego dwóch rejestrów przestawnych
4. Jak zmienić rozdzielczość odczytu temperatury w przedstawionym przykładzie 2.2.1
5. Czy istnieje inny sposób odczytu temperatury niż przedstawiony? Jeżeli tak – proszę go zaimplementować (zewnętrzna, dodatkowa biblioteka)
6. Zzaimplementować odczyt temperatury z większej ilości czujników (np.
7. . Czy tego typu rozwiązanie ma sens?
8. Zaimplementować przykład z PCF8574 wykorzystując przerwania. Układ należy rozbudować o wyświetlacz (podłączony np. do Arduino), dodatkowy przycisk (podłączony do Adruino) i 3 diody (ogólnie dwie mają być podłączone do rozszerzenia i dwie do Arduino). Główna pętla ma odliczać czas od uruchomienia (w pętli można zmieniać wyświetlany czas, natomiast sekundy odliczać licznikiem). W przypadku naciśnięcia jednego z przycisków jedna ze zmiennych ma zwiększać swoją wartość, a zmniejszać przy naciśnięciu drugiego (zakres liczbowy 0-1
9. . Diody mają reprezentować liczbę (ich ustawienie na płytce). Z kolei na wyświetlaczu ma wyświetlać się liczba w kodzie szesnastkowym. W celu ułatwienia zadania można wykorzystać gotową bibliotekę PCF8574.h (należy pobrać ją z internetu).
10. Zaimplementować obsługę komputera poprzez wykorzystanie pilota/telefonu z podczerwienią. Minimalnie pilot powinien pozwalać na uruchamianie przeglądarki, wylogowanie się z konta systemu oraz uruchomienie odtwarzacza filmów (domyślnego dla użytkowanego systemu operacyjnego).
11. Zaimplementować przemieszczanie po menu projektowa w sali (bez ciągłego włączania i wyłączania projektora; program ma nie mieć możliwości zmian opcji – jednie przemieszczanie się po nich).
12. Pracując w grupie (2 osoby – 2 zestawy) należy zaimplementować moduł do zdalnego sterowania komputerem. Moduł z jednej strony musi być wyposażony w emiter podczerwieni, przyciski sterujące (emulujące mysz) oraz podłączenie do klawiatury komputera. Z drugiej strony układ powinien posiadać odbiornik, wyświetlacz oraz podłączenie do komputera. Pierwszy układ ma transmitować sygnały emulacji myszy (klikanie poszczególnych przycisków) oraz przesyłać wszystkie kody klawiszy klawiatury. Z drugiej strony po każdorazowym odebraniu informacji na wyświetlaczu ma pojawiać się informacja czy emulowana jest mysz/klawiatura. Dodatkowo komputer ma wykonywać polecenia zadane przez układ zdalny. Realizacja powinna wykorzystywać jak najwięcej elementów poznanych na tych oraz poprzednich zajęciach. Przykładowo można dołożyć drugą parę nadawania podczerwieni (zwrotną), która będzie powodowała zapalanie się zielonej diody w przypadku wykonania zadania bądź czerwonej, jeżeli wystąpił błąd w wykonywaniu poleceń zdalnych. Aby wygenerować błędy układ odbiorczy można doposażyć w dodatkowy przycisk, który po wciśnięciu będzie blokował wykonywanie poleceń.

    DLA STUDENTÓW STUDIÓW NIESTACJONARNYCH:

13. Należy sporządzić sprawozdanie z przebiegu ćwiczenia według podanego szablonu (dostępny na stronie).

4 Materiały wykorzystane przy opracowaniu

https://www.arduino.cc/en/Tutorial/ShiftOut

https://github.com/Simsso/Shift-Register-74HC595-Arduino-Library

https://codebender.cc/sketch:296790#%5Barduino-tutorial%5D%2074HC595%20Shift%20Register%20IC.ino

http://cdn.instructables.com/FG9/72VS/IC37DCSQ/FG972VSIC37DCSQ.LARGE.jpg

https://www.rastating.com/using-a-74hc595-shift-register-with-an-arduino-uno/

https://create.arduino.cc/projecthub/TheGadgetBoy/ds18b20-digital-temperature-sensor-and-arduino-

http://akademia.nettigo.pl/ds18b20/

https://botland.com.pl/index.php?controller=attachment&id_attachment=26

http://akademia.nettigo.pl/ds18b20/DS18B20.zip

http://files.arduino-projekte.webnode.at/200000916-4597f46953/PCF8574%20Test_Steckplatine.jpg

http://arduino-projekte.webnode.at/projekte/portexpander-pcf8574/

http://starter-kit.nettigo.pl/2011/11/pcf8574-czyli-jak-latwo-zwiekszyc-liczbe-pinow-w-arduino/

http://akademia.nettigo.pl/starter_kit_030/

https://majsterkowo.pl/ekspander-pcf8574/

https://www.arduino.cc/en/reference/wire

http://openlgtv.org.ru/wiki/index.php/LG_TV_USB_IR-Hack_with_Arduino

https://majsterkowo.pl/jak-sterowac-dowolnym-urzadzeniem-za-pomoca-pilota-i-arduino/

https://github.com/z3t0/Arduino-IRremote/blob/master/IRremote.h