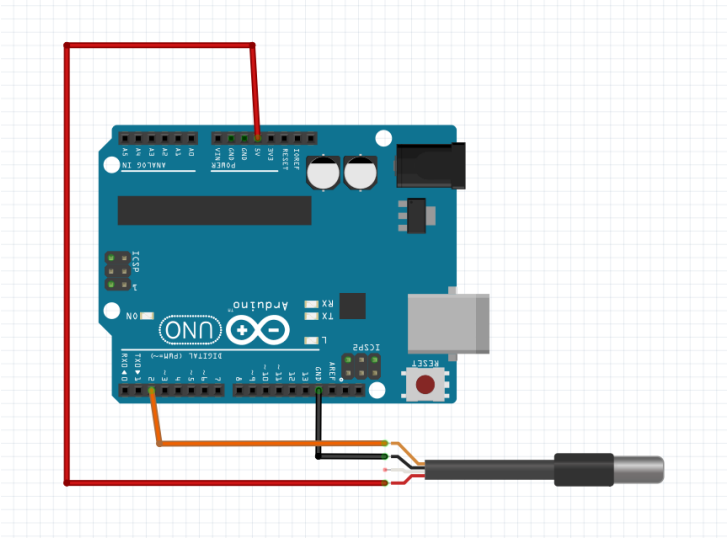
Laboratorium z przedmiotu Systemy wbudowane (SW)

Zadanie nr 2

Temat zajęć: Arduino - elementy pomiarowe

Prowadzący	Autorzy	Grupa dziekańska:
mgr inż. Ariel Antonowicz	148088 i 148121	I1.2

1 Termometr (DS18B20) – One wire



Schemat podłączenia termometru do Arduino

1.1 Opis działania OneWire

OneWire jest interfejsem komunikacyjnym. Nazwa wynika z tego, że do komunikacji między urządzeniami wyposażonymi w ten interfejs wystarczy jedna linia transmisyjna (nie licząc masy). OneWire ma strukturę Master-Slave, co oznacza, że urządzenie nadrzędne (Master) wysyła komendy do urządzeń podrzędnych (Slave), których może być wiele. W celu odróżnienia slave od siebie, nadaje się im unikatowe 64-bitowe numery seryjne.

Komunikacja na linii danych przebiega w dwie strony, z tego powodu dołącza się do magistrali bufory trójstanowe lub bramki z otwartym drenem, w celu utrzymania porządku wysyłania danych.

Master rozpoczyna transmisję danych poprzez wystawienie impulsu reset. To powoduje zresetowanie wszystkich podłączonych urządzeń podrzędnych, które następnie wysyłają impuls obecności na linię danych. Od tego momentu, master może przesyłać sekwencje bitów do slave'ów. Urządzenie podrzędne wykona określone akcje, tylko w przypadku gdy dane polecenie dotyczy tego urządzenia. Master odczytuje dane z urządzeń slave.

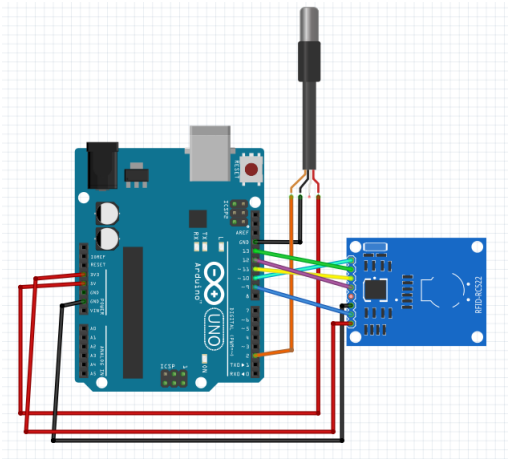
```
#include <OneWire.h>
const byte ONEWIRE_PIN = 2;
OneWire onewire(ONEWIRE_PIN);
DS18B20 sensors(&onewire);

void setup(){
  while(!Serial);
  Serial.begin(9600);
}

void loop(){
  byte address[8];
  onewire.reset_search();
  while(onewire.search(address))
  {
    if (address[0] != 0x28)
    continue;

    if (OneWire::crc8(address, 7) != address[7]){
      Serial.println(F("Bledny adres, sprawdz polaczenia"));
      break;
    }

    for (byte i=0; i<8; i++){
      Serial.print(F("0x"));
      Serial.print(address[i], HEX);

      if (i < 7)
      Serial.print(F(", "));
    }
    Serial.println();
  }
  while(1);
}
// wypisany na Serial Monitor adres:
// 0x28, 0xFF, 0xBC, 0x88, 0x90, 0x17, 0x5, 0x76

```

Kod użyty do wyznaczenia adresu czujnika

```
#include <OneWire.h>
#include <DS18B20.h>
#include <SPI.h>
#include <MFRC522.h>

#define ONEWIRE_PIN 2

// Adres czujnika
byte address[8] = {0x28, 0xFF, 0xBC, 0x88, 0x90, 0x17, 0x5, 0x76};

OneWire onewire(ONEWIRE_PIN);

int sort_desc(const void *cmp1, const void *cmp2){
  int a = *((int *)cmp1);
  int b = *((int *)cmp2);
  return a > b ? -1 : (a < b ? 1 : 0);
}

void setup() {
  while(!Serial);
  Serial.begin(9600);
  SPI.begin();       // Initiate  SPI bus
  sensors.begin();
  sensors.request(address);
}

void loop() {
  float vec[20];
  if (sensors.available())
  {
    for (int i = 0; i < 18; i++) {
      vec[i] = sensors.readTemperature(address);
    }
    qsort(vec, 18, sizeof(float), sort_desc);

    float sum = 0;
    for (int i = 1; i < 17; i++) {
      sum += vec[i];
    }
    sum /= 16;

    Serial.print(sum);
    Serial.println(F(" 'C"));
    sensors.request(address);
  }
}

```

Kod użyty do wyznaczenia średniej temperatury

2 RFID



Schemat podłączenia RFID do Arduino

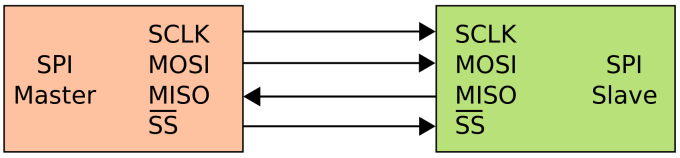
```
#include <OneWire.h>
#include <DS18B20.h>
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance.
#define ONEWIRE_PIN 2

// Adres czujnika
byte address[8] = {0x28, 0xFF, 0xBC, 0x88, 0x90, 0x17, 0x5, 0x76};
OneWire onewire(ONEWIRE_PIN);
DS18B20 sensors(&onewire);

int sort_desc(const void *cmp1, const void *cmp2){
  int a = *((int *)cmp1);
  int b = *((int *)cmp2);
  return a > b ? -1 : (a < b ? 1 : 0);
}

void setup() {
  while(!Serial);
  Serial.begin(9600);
  SPI.begin();        // Initiate  SPI bus
  mfrc522.PCD_Init();   // Initiate MFRC522
  sensors.begin();
  sensors.request(address);
}

void loop() {
  float vec[20];
  if (sensors.available()){
    for (int i = 0; i < 18; i++) {
      vec[i] = sensors.readTemperature(address);
    }
    qsort(vec, 18, sizeof(float), sort_desc);

    float sum = 0;
    for (int i = 1; i < 17; i++) {
      sum += vec[i];
    }
    sum /= 16;

    if ( ! mfrc522.PICC_IsNewCardPresent()) return;
    if ( ! mfrc522.PICC_ReadCardSerial()) return;

    String content= "";
    byte letter;
    for (byte i = 0; i < mfrc522.uid.size; i++){
      content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
      content.concat(String(mfrc522.uid.uidByte[i], HEX));
    }
    // adresy odczytane dzieki powyzszej petli:
    // C4 F7 2E D0 // brelok
    // 6D 3D AC 75 // karta
    content.toUpperCase();
    if (content.substring(1) == "BD 31 15 2B"){
      Serial.print(sum);
      Serial.println(F(" 'C"));
    }
    else{
      Serial.println("Brak dostepu");
    }
    sensors.request(address);
  }
}
```

Kod źródłowy wyświetlający temperature po przyłożeniu karty do modułu RFID

2.1 Opis działania SPI

SPI (Serial Peripheral Interface) to interfejs komunikacyjny pozwalający na komunikację full-duplex, czyli można jednocześnie wysyłać i odbierać dane. SPI jest również interfejsem synchronicznym - jeden z przewodów przesyła sygnał zegarowy, który synchronizuje wszystkie urządzenia podrzędne. Prędkość transmisji dostosowywana jest do najwolniejszego ze stosowanych układów.

Tak samo jak w interfejsie OneWire, SPI też ma strukturę typu Master-Slave. Również w tym przypadku, możliwe jest połączenie kilku układów slave, dołącza się bufor trójstanowy oraz tylko master może rozpocząć transmisję. Dodatkowo, master może generować sygnał zegarowy.



Przykładowa konfiguracja układów master oraz slave w systemie.

Magistrala SPI składa się najczęściej z czterech linii. Są to:

- MOSI (master out, slave in) – linia łącząca wyjście danych z mastera i wejścia slave'ów
- MISO (master in, slave out) – linia łącząca wyjście danych slave i wejście mastera
- SCLK (serial clock) – sygnał zegarowy, synchronizujący układy na magistrali
- CS (chip select), SS (slave select) – pozwala na wybranie układu, który będzie przeprowadzał transmisję

Aby rozpocząć transmisję, master musi wysłać sygnał zegarowy i wybrać układ, ściągając do masy odpowiednią linię CS. Linie MISO i MOSI tworzą pierścień, przez co układy slave i master mogą jednocześnie wysyłać dane do siebie nawzajem. Dane są synchronicznie przenoszone na linię MOSI i czytane z linii MISO.

Źródła

1. Fritzing
2. RFID DS18B20
3. Materiały podane przez prowadzącego na platformie ekursy.
4. randomnerdtutorials
5. 1-Wire RS Elektronika
6. Interfejs SPI - Extronic
7. Wprowadzenie do interfejsu SPI - elektroda

Contents

1 Termometr (DS18B20) – One wire **1**
1.1 Opis działania OneWire 1

2 RFID **3**
2.1 Opis działania SPI 4