Technika Mikroprocesorowa 2

Adam Gajek, Kamil Kulak
Środa, 9:35
2014/2015

1. Cel projektu

Celem projektu jest zaprogramowanie mikrokontrolera spełniającego funkcjonalność zdalnego zarządcy podłączonych zasobów. Gotowy projekt będzie mógł zostać wykorzystany m.in. jako sterownik w systemach “inteligentnych domów”.

2. Zamierzone funkcjonalności

Główną funkcjonalnością będzie możliwość informowania za pośrednictwem modemu GSM o zajściu konkretnego zdarzenia, a także reagowania na zdefiniowane informacje przychodzące poprzez wyżej wymieniony modem.

Przykładowym zdarzeniem w naszym prototypem będzie informowanie o przekroczenia temperatury otoczenia mierzonej za pomocą cyfrowego termometru. Przewidywana jest możliwość konfiguracji wartości progowej temperatury po przekroczeniu której wysyłana jest wiadomość SMS.

Zapewniamy także możliwość rozszerzania komunikacji z innymi podłączonymi urządzeniami. W modelu prototypowym możliwości takiego rozszerzenia będziemy prezentować poprzez zapalające się diody na odpowiednich wejściach po odebraniu odpowiedniego sygnału przez modem GSM.

Planowane jest użycie następujących części:

- Arduino Leonardo

- Siemens S45(moduł GSM)

- Dallas DS18B20

Podsumowanie planowanej funkcjonalności:

- ustawianie granicznej wartości temperatury na czujniku Dallas za pomocą wiadomości SMS na dany numer telefonu, po przekroczeniu której, odbiorca/odbiorcy, będą informowaniu o zdarzeniu(również za pomocą wiadomości SMS);
- umożliwienie włączenia/wyłączenia diody LED na Arduino, za pomocą wiadomości SMS;
- uruchomienie zdarzeń cyklicznych za pomocą wiadomości SMS(w formie prostego schedulera, np włączenie/wyłączenie diody LED co 5 min).

3. Harmonogram prac

Data	Planowany zakres prac
20.10.2014	● “Research” dotyczący wykonania projektu, potrzebnych elementów, użytych technologii, ● Stworzenie szkieletu dokumentacji,
05.11.2014	● Skompletowanie niezbędnych części,
12.11.2014	● Uruchomienie przykładowego projektu(obejmującego uruchomienie czujnika na gotowych bibliotekach, sprawdzenie poprawności wskazywanej temperatury z rzeczywistą),

10.12.2014	● Implementacja obsługi interfejsu 1-Wire, ● Wykonanie testowych odczytów temperatury z czujnika,
17.12.2014	● Podłączenie modułu GSM(telefonu) do mikrokontrolera, ● Implementacja obsługi modułu GSM, ● Wykonanie testowych wiadomości SMS,
07.01.2015	● Implementacja pełnej funkcjonalności, ● Uzupełnienie dokumentacji, oddanie końcowego projektu.

4. Zakres wykonanych prac

- research dot. wykonania projektu
- skompletowanie wszystkich niezbędnych części
- uruchomienie przykładowego projektu poprzez wykorzystanie gotowych blibliotek Arduino w celu sprawdzenia połączeń i konfiguracji Arduino i czujnika temperatury
- implementacja obsługi interfejsu 1-Wire + czujnika Dallas(stworzenie biblioteki do odczytu temperatury)
- testy dot. odczytu temperatury
- skompletowanie brakującej części - kabla do podłączenia telefonu, próba komunikacji (nieudana :( )
- podłączenie telefonu do Arduino za pośrednictwem łącza szeregowego
- zebranie wiadomości nt ramek PDU i komend AT
- wysłanie oraz odebranie przykładowego SMSa z Arduino za pośrednictwem podłączonego telefonu,
- zaimplementowanie konwertera wiadomości tekstowych do formatu PDU wymaganego przez komenty AT,
- automatyczne wysyłanie SMSów po przekroczeniu progowej temperatury,
- konfiguracja progowej temperatury za pośrednictwem SMSów,
- kontrola nad diodami LED za pośrednictwem SMSów,

4. Opis technologii

a. **komendy AT** - w użytym przez nas telefonie znajduje się modem GSM, który umożliwia komunikację przy użyciu seryjnego portu za pośrednictwem komend AT. Każda komenda rozpoczyna się od słowa AT po którym występują argumenty komendy. Użyte zostały następujące komendy:

- AT+CMGF=<mode><CR> - przygotowanie do wysłania wiadomości SMS
  <mode> - tryb wysyłania SMS: 0 - tryb PDU, 1 - tryb tekstowy
  <CR> - znak powrotu karetki kursora
- AT+CMGS=<length><CR><PDU><CTRL+Z> - wysyłanie wiadomości SMS
  <length> - długość wiadomości <PDU>
  <CR> - znak powrotu karetki kursora
  <PDU> - zakodowana wiadomość w formacie PDU(zawiera również nr tel odbiorcy)
  <CTRL+Z> - CTRL+Z
- AT+CMGL=<mode><CR> - przygotowanie do odczytu wiadomości SMS. Argumenty jak wyżej

b. **ramka PDU** - wiadomości SMS w sieci GSM są przesyłane jako ramki PDU. Jest to format wykorzystywany w telekomunikacji w celu przesyłania treści wraz ze wszystkimi niezbędnymi informacjami jako spójna jednostka. Informacja zawarta w ramce PDU jest zakodowana w formacie heksadecymalnym. Poniżej opiszę poszczególne części dla przykładowej ramki:

07917283010010F5040BC87238880900F10000993092516195800AE8329BFD4697D9EC37

07 - liczba oktetów, które kodują informacje dotyczące centrum SMS
91 - typ SMSC (91 - oznacza międzynarodowy format)
72 83 01 00 10 F5 - numer SMSC. W tym przypadku jest to: +27 381 000 015

jednak wymaga on małej konwersji, która polega na podzieleniu numeru telefonu na dwu cyfrowe części i dopisaniu F w przypadku nieparzystej liczby cyfr, a następnie zamianie kolejności w poszczególnych segmentach. Stąd 27 zmieniamy na 72, a 38 na 83.

04 - wartość ta interpretowana jest jako SMS-DELIVER - jest to tryb pobierania wiadomości z SMSC
0B - długość numeru SMSC jako wartość heksadecymalna
C8 - typ adresu nadawcy wiadomości

72 38 88 09 00 F1 - numer nadawcy w formacie takim jak numer SMSC
00 - identyfikator protokołu
00 - schemat kodowania wiadomości tekstowej. 00 oznacza kodowanie 8 bitowych danych na 7 bitowym alfabecie w celu kompresji rozmiaru wiadomości
99309251619580 - data wysłania widomości
0A - długość treści wiadomości jako wartość heksadecymalna
E8329BFD4697D9EC37 - treść wiadomości reprezentowana w postaci 8-bit oktetów reprezentujących 7 bitowe dane

c. **kodowanie i dekodowanie treści wiadomości -** po przekształceniu do postaci binarnej należy usunąć początkowe zera.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
01010011	**01001101**	**01010011**	**00100000**	**01010010**	**01110101**	**01101100**	**01111010**

Następnie idąc od początku w pierwszej iteracji zabieramy najstarszy bit z drugiego bajtu i dokładamy na początek pierwszego, następnie zabieramy dwa najstarsze bity trzeciego bajtu i dokładamy na początek drugiego bajtu.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
11010011	**11100110**	**10100xx**	**0100000**	**1010010**	**1110101**	**1101100**	**1111010**

Wykonujemy tę operację do czasu kiedy dotrzemy do siódmegu bajtu do początku którego dokładamy całość ósmego bajtu.

1	2	3	4	5	6	7	8
S	**M**	**S**		**R**	**u**	**l**	**z**
53	**4D**	**53**	**20**	**52**	**75**	**6C**	**7A**
11010011	**11100110**	**00010100**	**00100100**	**10101101**	**10110011**	**11110101**	**xxxxxxx**

Jeśli napis był dłuższy niż 8 znaków powtarzamy tę operację dla kolejnej części.

Dekodowanie jest operacją odwrotną

d. Interfejs 1-Wire

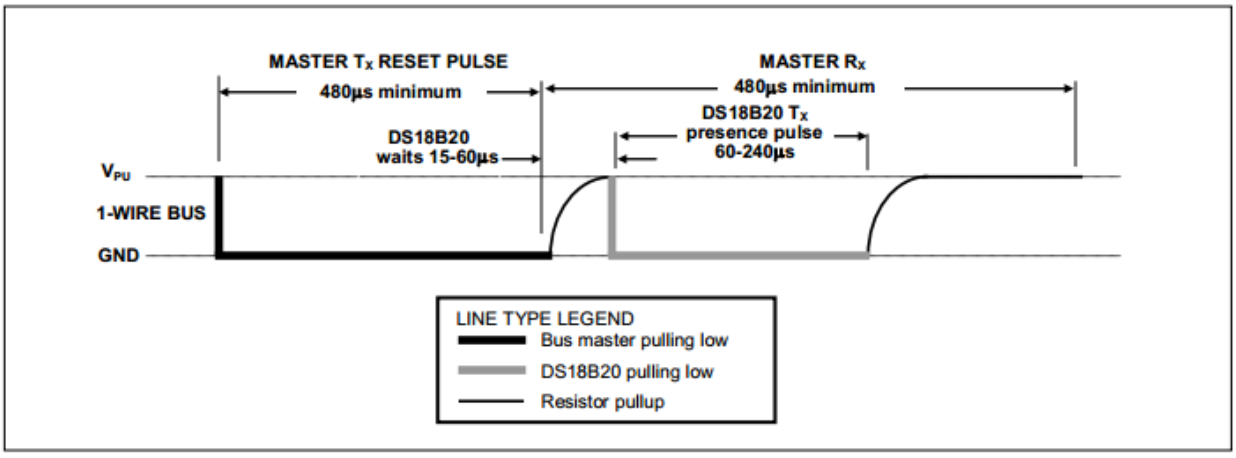
1-Wire jest protokołem komunikacyjnym pomiędzy urządzeniami, którego nazwa wywodzi się z faktu, że do komunikacji używa jest tylko jedna linia danych(+GND). Odbiornik może być zasilany bezpośrednio z linii danych(mówimy wtedy o zasilaniu pasożytniczym), lub wykorzystywać osobny przewód. W naszym projekcie, 1-Wire jest wykorzystywany do komunikacji z czujnikiem temperatury Dallas DS18B20.

Zasady transmisji danych

Przesłanie każdej porcji danych inicjalizowane jest przez Reset Pulse wysyłany przez układ Master, oraz Presence Pulse przez Slave'a. Zadaniem Reset Pulse'u jest zresetowanie wszystkich podłączonych odbiorników, natomiast Presence Pulse jest potwierdzeniem gotowości urządzeń typu Slave'a na komunikację. Reset Pulse reprezentowany jest poprzez ustawienie linii danych w stan niski na min 480us. Następnie pull-up rezystor podciągnie linię danych w stan wysoki, po czym urządzenie Slave ustawi ponownie linię danych w stan niski na okres 60-240us(Presence Pulse).

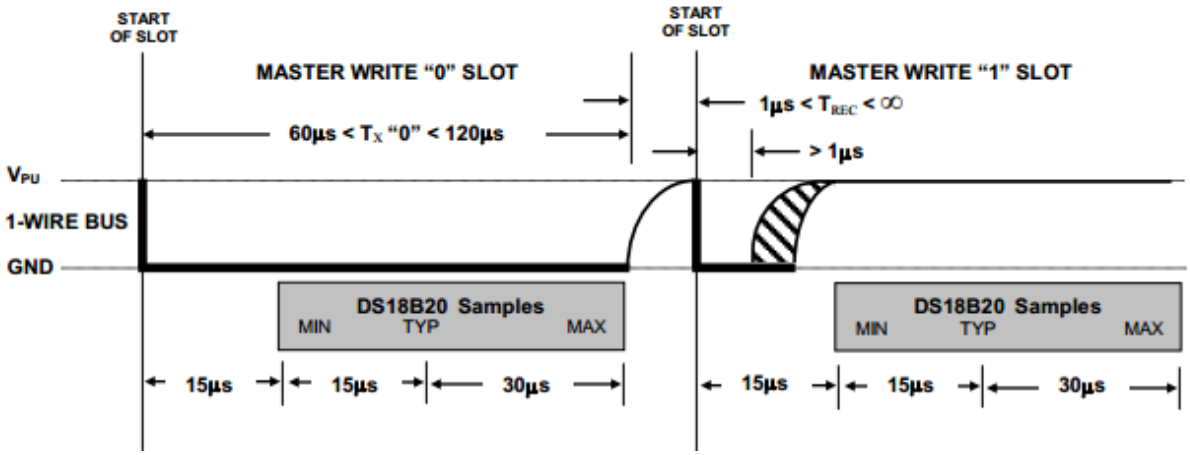
Schemat przedstawiony jest poniżej:

Figure 13. Initialization Timing

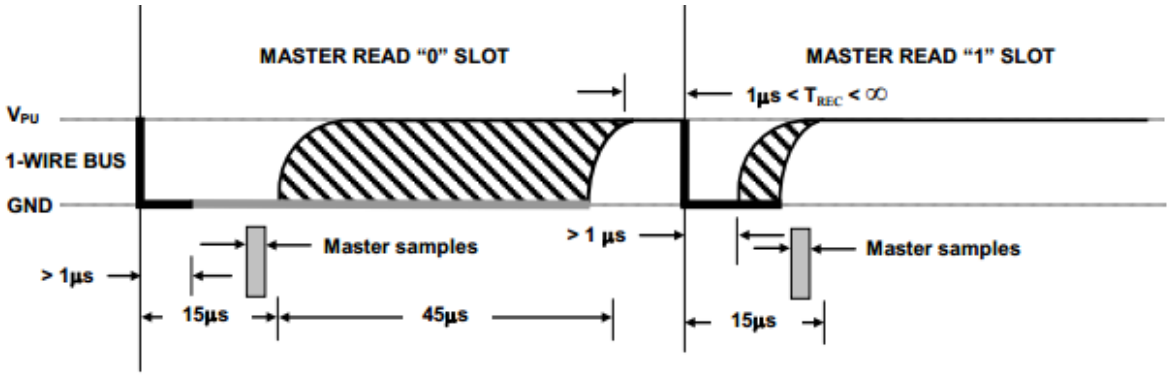


Wysłanie bitu informacji do układu Slave polega na ustawieniu przez Mastera linii danych w stan niski na krótki okres czasu(1-15us). Jeśli wysyłamy '1', ustawiamy linię danych w stan wysoki(np poprzez rezystor podciągający), w przeciwnym wypadku, pozostawiamy stan niski. Przedziały te powinny trwać min 60us.

Schemat przedstawiony jest poniżej:



Odebranie bitu informacji od Slave'a polega na ustawieniu przez Mastera linii danych na poziom niski przez 1-15us, a następnym powrocie stanu linii danych do poziomu wysokiego. Jeśli układ Slave'a będzie chciał przesłać '1', nie robi nic, w przeciwym przypadku ustawia linię danych na '0' na 60us.
Schemat przedstawiony jest poniżej:



Mając już zdefiniowane funkcję do odczytu oraz przesłania informacji z DS18B20 możemy zająć się odczytem temperatury. Cała procedura upraszcza się do wysłania oraz odebranie kilku bajtów:

1. Wysyłamy RESET PULSE.
2. Na naszej magistralii podłączone jest tylko jedno urządzenie typu Slave, dlatego możemy ominąć proces odpytywania o adres ROM urządzenia Slave, przy pomocy komendy SKIP ROM(0xCC).
3. Włączamy konwersję temperatury, ConvertT(0x44). Może to potrwać do 750ms.
4. Ponownie wysyłamy RESET PULSE oraz SKIP ROM.
5. Odpytujemy o odczytaną wartość temperatury, poprzez wysłanie komendy odczytania zawartości scratchpada(0xBE).

6. Po odczycie dwóch pierwszych bajtów(LSB, MSB) kończymy odczyt, dokonujemy konwersji, oraz otrzymujemy ostateczny wynik

5. Implementacja

Dostępna jest na repozytorium BitBucket oraz dołączona jako osobna paczka do dokumentacji. Link:
https://bitbucket.org/tesladev/microprocessor-project

6. Instrukcja użytkowania

Kontrolę nad mikroprocesorem sprawujemy przy pomocy komend wysyłanych poprzez wiadomości SMS na numer telefonu 606 588 366, który służy jako modem GSM zapewniający bezprzewodową obsługę mikroprocesora.

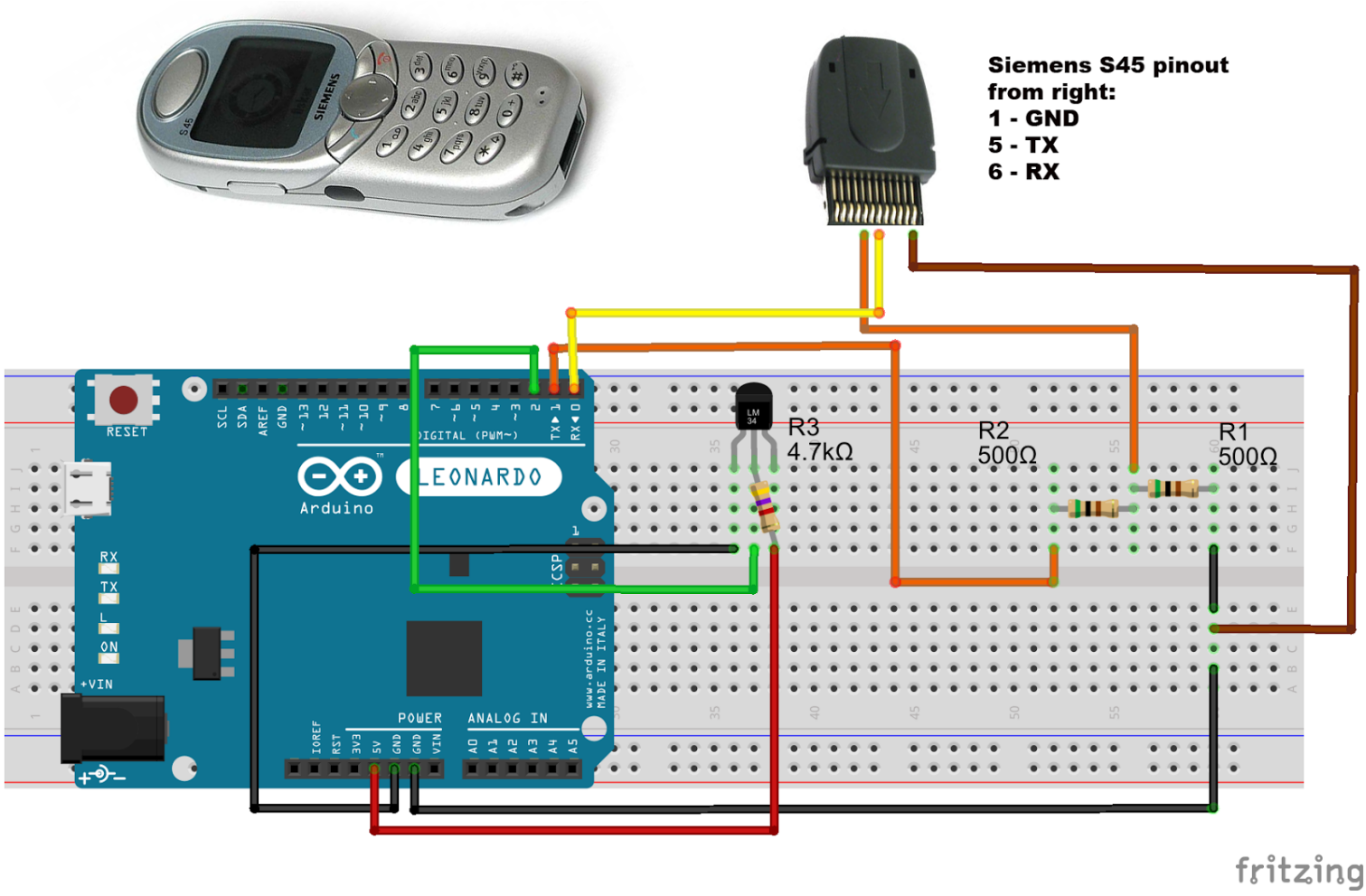
Aby skonfigurować graniczną temperaturę, po przekroczeniu której mikroprocesor poinformuje poprzez wysłanie wiadomości SMS należy wysłać SMS o treści **T <temp>** gdzie
<temp> to wartość graniczna temperatury w stopniach Celsjusza.

Aby spowodować zaświecenie się diody LED na płytce należy wysłać SMS o treści **on led**. Jeśli chcemy diodę dezaktywować treścią wiadomości SMS będzie **off led.**

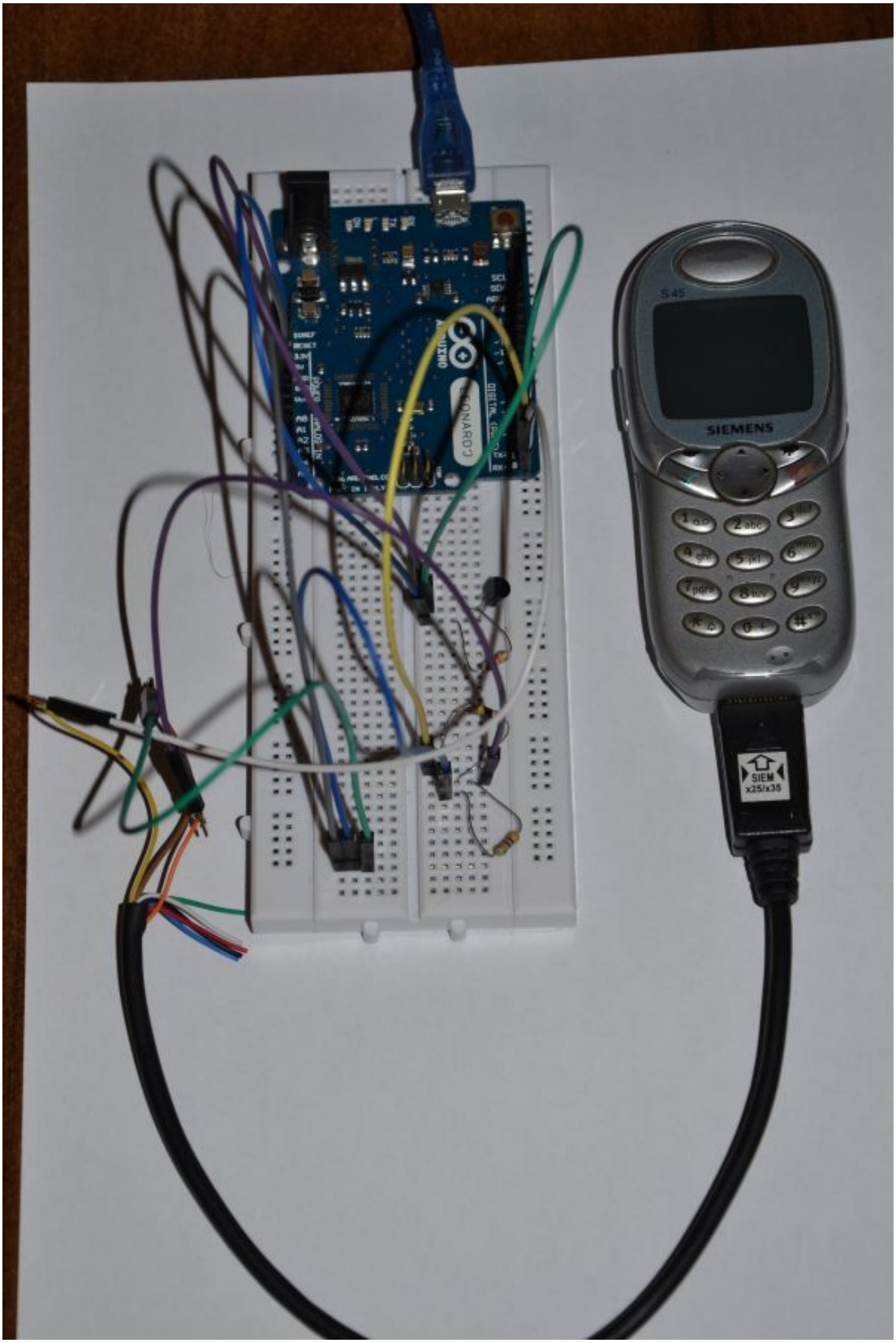
Aby uruchomić cykliczne wydarzenie reprezentowane w naszym prototypie jako cykliczne zaświecanie i gaszenie diody LED należy wysłać SMS o treści
tg <period> <repeat count>, gdzie
<period> to okres trwania zdarzenia[s],
<repeat count> to ilość cykli

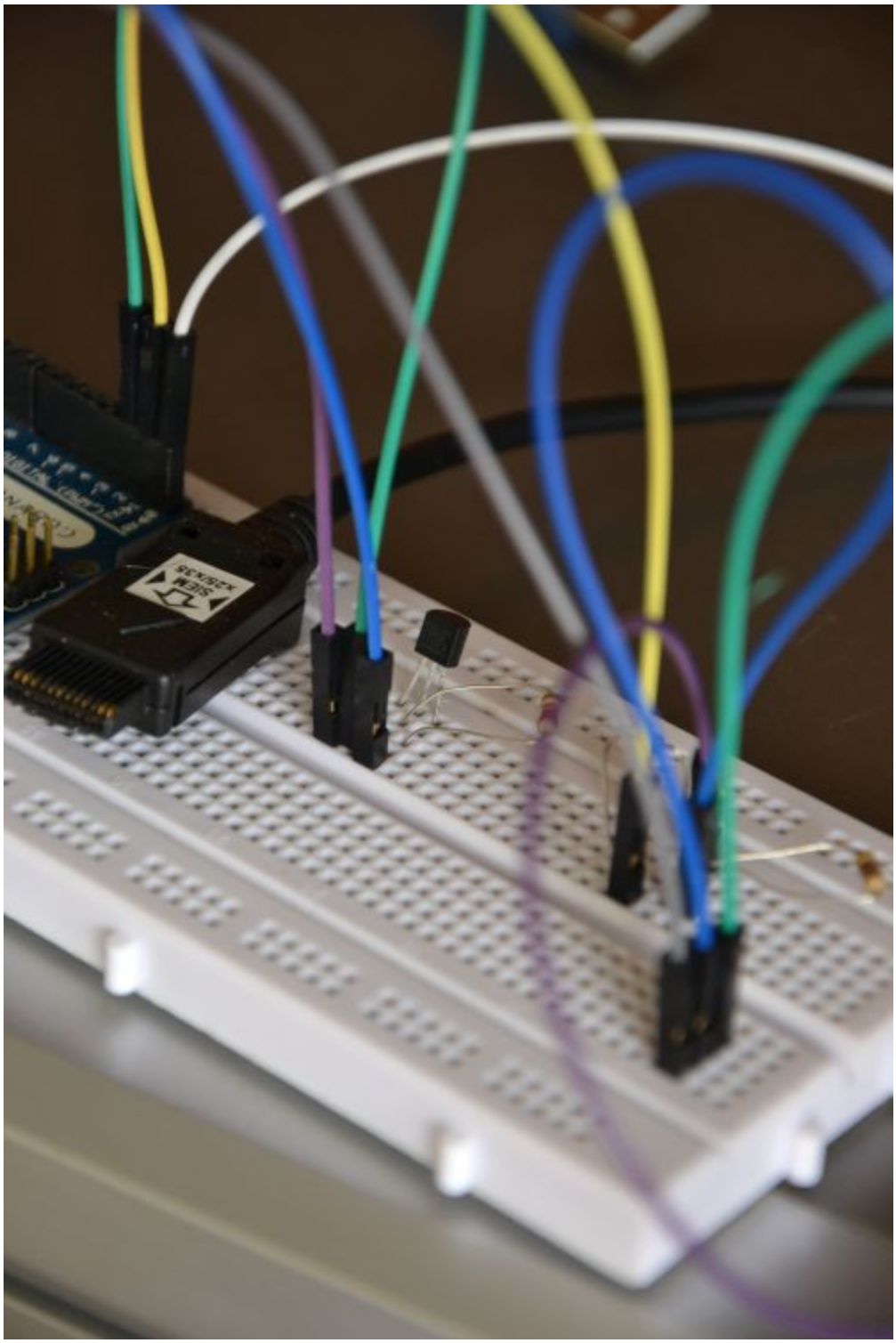
W razie potrzeby istnieje możliwość obsługi dowolnego urządzenia po podpięciu do konkretnego portu Arduino i zdefiniowaniu komend, które będą służyły do obsługi danego urządzenia.

7. Schemat


Siemens S45 pinout
from right:
1 - GND
5 - TX
6 - RX
R3
4.7kΩ
R2
500Ω
R1
500Ω
LEONARDO
Arduino
RESET
fritzing

8. Zdjęcia projektu





9. Bibliografia

1. http://arduino.cc/en/Main/arduinoBoardLeonardo
2. http://te.ugm.ac.id/~enas/tpe/temu5/GPRS_AT_CommandSet.pdf
3. http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
4. http://en.wikipedia.org/wiki/1-Wire
5. http://rednaxela.net/pdu.php
6. http://forum.arduino.cc/index.php?topic=67914.0
7. http://mobileforensics.files.wordpress.com/2007/06/understanding_sms.pdf - kodowanie i dekodowanie treści wiadomości